



UNIVERSIDADE ESTADUAL PAULISTA
“JÚLIO DE MESQUITA FILHO”
Câmpus de São José do Rio Preto

Mariele Pedro de Souza

Classificação de imagens tridimensionais por meio de
homologia persistente e imagem de persistência

São José do Rio Preto
2022

Mariele Pedro de Souza

Classificação de imagens tridimensionais por meio de
homologia persistente e imagem de persistência

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Orientador: Prof. Dr. Thiago de Melo

Financiadora: CNPq

São José do Rio Preto
2022

S729c

Souza, Mariele Pedro de

Classificação de imagens tridimensionais por meio de homologia persistente e imagem de persistência. / Mariele Pedro de Souza. -- São José do Rio Preto, 2022

111 p.

Dissertação (mestrado) - Universidade Estadual Paulista (Unesp), Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto

Orientador: Thiago de Melo

1. Matemática. 2. Topologia algébrica. 3. Análise topológica de dados. 4. Homologia persistente. 5. Agrupamento. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca do Instituto de Biociências Letras e Ciências Exatas, São José do Rio Preto. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Mariele Pedro de Souza

Classificação de imagens tridimensionais por meio de
homologia persistente e imagem de persistência

Dissertação apresentada como parte dos requisitos para obtenção do título de Mestre em Matemática, junto ao Programa de Pós-Graduação em Matemática, do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista “Júlio de Mesquita Filho”, Câmpus de São José do Rio Preto.

Financiadora: CNPq

Comissão Examinadora

Prof. Dr. Thiago de Melo
Orientador

Profa. Dra. Evelin Meneguessso Barbaresco
Departamento de Matemática - Unesp São José do Rio Preto

Prof. Dr. Northon Canevari Leme Penteado
Departamento de Matemática - Universidade Federal do Cariri

São José do Rio Preto
08 de setembro de 2022

À minha brasilidade de nunca desistir.

AGRADECIMENTOS

Muitos se ufanam: “Não devo nada a ninguém.”

Engano: devemos muito a todos.

(CORALINA, Cora).¹

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelos vinte e quatro meses de bolsa concedida, por meio do processo número 132281/2020-1. Ajuda essencial em um período de horrores vivido pela pandemia.

Agradeço aos que estiveram comigo por todos estes longos meses, que contribuíram à sua maneira para que este trabalho fosse realizado. À minha família, pelo privilégio financeiro e por aceitarem a minha dedicação. Aos meus irmãos, em particular, que sempre me lembravam que “só estudar” significava muito mais que isso. Aos grandes amigos que a matemática me deu, que entendiam o meu desespero e me encorajaram a concluir mais este grau acadêmico (Mateus e Maurício, vocês foram essenciais).

Agradeço à Professora Doutora Évelin Meneguesso Barbaresco por ter me acolhido pacientemente, me apresentado à Topologia e estar disposta a me acompanhar em algo novo.

Ao meu orientador, Professor Doutor Thiago de Melo, toda minha gratidão pelos ensinamentos, por me acompanhar durante estes meses e entender o meu tempo.

¹Disponível em: <https://poemasfrasesemensagens.wordpress.com/>.

O conhecimento serve para encantar pessoas, não para humilhá-las.
(CORTELLA, Mario Sergio). [6]

RESUMO

O trabalho desenvolvido traz um estudo sobre o agrupamento de imagens tridimensionais por meio da Análise Topológica de Dados (TDA). Nos primeiros capítulos, as principais definições da Topologia Algébrica são apresentadas para construção e entendimento dos cálculos realizados posteriormente. À seguir, são desenvolvidos programas na linguagem computacional `Python` para o estudo das mais de cento e cinquenta imagens, de vinte categorias distintas. Os agrupamentos foram realizados segundo a distância entre diagramas de persistência e imagens de persistência. As análises finais foram realizadas por meio de escalonamento multidimensional, dendrogramas e algoritmo k-means.

Palavras-chave: Topologia. Análise topológica de dados. Diagrama de persistência. Agrupamento.

ABSTRACT

The work developed brings a study on the clustering of three-dimensional images through the Topological Data Analysis (TDA). In the first chapters, the main definitions of Algebraic Topology are presented for the construction and understanding of the calculations performed later. Next, programs are developed in the computer language Python for the study of more than one hundred and fifty images, from twenty different categories. Groupings were performed according to the distance between persistence diagrams and persistence images. Final analyses were performed using multidimensional scaling, dendrograms, and the k-means algorithm.

Keywords: Topology. Topological data analysis. Persistence diagram. Clustering.

Lista de Figuras

2.1	Pontos Geometricamente dependentes e independentes.	22
2.2	Exemplos de k -simplexos	22
2.3	Tipos de face do σ^2	23
2.4	Simplexos propriamente unidos e uma união imprópria	24
2.5	Realização geométrica.	26
2.6	2-simplexo orientado	27
2.7	Exemplo de 1-simplexo e 2-simplexo	28
3.1	Complexo de Čech	31
3.2	Complexo VR	32
3.3	Diagrama de Voronoi	33
3.4	Complexo Alpha	34
3.5	Filtração de um complexo simplicial	34
3.6	Nascimento e morte na sequência de grupos de homologia	36
3.7	Complexo simplicial filtrado.	37
3.8	Barcode e Diagrama de Persistência	37
3.9	Diagrama de Persistência H_0	41
3.10	Gaussiana de uma Imagem de Persistência	44
4.1	Exemplos de imagens do banco de dados.	46
4.2	Imagens com quantidade de pontos diferentes.	47
4.3	PDs de imagens com diferente quantidade de pontos	51
4.4	Diagrama de Persistência do Cubo.	52
4.5	Escala de cor utilizada para colorir as imagens.	53
4.6	Table 3, 56 pontos.	55
4.7	Potted Plant 8, 1.325 pontos.	55
4.8	Head 5, 2.034 pontos.	56
4.9	Fighter Jet 1, 4.427 pontos.	56
4.11	Matriz de Distância Bottleneck	58
4.12	Distância Bottleneck de até 0,01	61
4.14	Imagem próxima à Fighter jet 9.	62
4.15	Distância Bottleneck entre Fighter Jet 9 e Biplane 6.	63
4.16	Table 0, 60 pontos.	66
4.17	Cubo - untitled3, 8 pontos.	67
4.18	Matriz de Distância Euclidiana	70
4.19	Matriz de Distância Euclidiana com peso	71
4.20	Distância Euclidiana de até 0.01	72
4.21	Distância Euclidiana com peso de até 0.0001	72
4.23	Imagem próxima à Biplane 4	73
4.25	Imagem próxima à Helicopter 8	74

4.26	Matriz de Distância do Máximo	75
4.27	Matriz de Distância do Máximo com peso	76
4.28	Distância do Máximo de até 0.001	76
4.29	Distância do Máximo com peso de até 0.00001	76
4.31	Imagem próxima à Deskchair 1	77
4.33	Imagem próxima à Fish 5	78
4.34	Matriz de Distância da Soma	79
4.35	Matriz de Distância da Soma com peso	79
4.36	Distância da Soma de até 0.1	80
4.37	Distância da Soma com peso de até 0.001	80
4.39	Imagem próxima à Fish 2	81
4.41	Imagem próxima à Flying Bird 4	82
5.1	MDS distância bottleneck.	86
5.2	MDS distância bottleneck de 5 categorias.	86
5.3	MDS distância euclidiana com peso de 5 categorias.	87
5.4	MDS distância do máximo com peso de 5 categorias.	87
5.5	MDS distância da soma com peso de 5 categorias.	87
5.6	Exemplo de dendrograma.	88
5.7	Dendrograma truncado da matriz de distância bottleneck.	90
5.8	Dendrograma truncado da matriz da soma.	91
5.9	Dendrograma truncado da Distância da Soma com peso	91
5.10	Dendrograma horizontal da Distância Bottleneck	92
5.11	Dendrograma da Distância da Soma com peso	93
6.1	Folhas verdes do dendrograma 5.10.	103
6.2	Folhas verdes do dendrograma 5.11.	104

Lista de Programas

4.1	Programa para gerar as matrizes de pontos e armazenar as informações. . .	48
4.2	Função que calcula H_1 de um arquivo, por meio de sua matriz de pontos. .	49
4.3	Programa para gerar os PDs das imagens com até 1.000 pontos.	49
4.4	Função para exibir o diagrama de persistência de uma imagem.	50
4.5	Função para gerar o diagrama de persistência (nascimento, persistência) e imagem de persistência com peso.	54
4.6	Função para gerar a imagem de persistência.	55
4.7	Calcular a distância bottleneck entre os Diagramas de Persistência. . . .	58
4.8	Função para tornar uma matriz simétrica quando precisa excluir última coluna vazia.	59
4.9	Função para inserir nome das colunas e índices, em uma matriz.	59
4.10	Função para filtrar as imagens mais próximas.	60
4.11	Função que exibe a nuvem de pontos das imagens filtradas.	62
4.12	Função para calcular a distância Bottleneck entre dois diagramas de per- sistência e gerar o gráfico.	63
4.13	Encontrar tamanho ideal para intervalo de nascimento e persistência. . . .	65
4.14	Função para calcular distância entre as matrizes das imagens de persistência.	69
5.1	Função para separar os arquivos em categorias.	84
5.2	Função para criar coluna de categorias nos <code>csv</code> 's.	85
5.3	Função para gerar MDS de uma matriz de distâncias.	85
5.4	Função para gerar dendrogramas de uma matriz de distâncias.	90
5.5	Função para calcular o agrupamento K -means.	98

Sumário

1	Introdução	19
2	Conhecimento preliminar	21
3	Diagrama de Persistência	31
3.1	Complexos	31
3.1.1	Complexo de Čech	31
3.1.2	Complexo de Vietoris-Rips (VR)	32
3.1.3	Complexo Alpha	33
3.1.4	Filtração	34
3.2	Homologia Persistente	35
3.2.1	A matriz bordo	38
3.3	Distância Bottleneck	40
3.4	Imagem de Persistência	43
4	Trabalho Desenvolvido	45
4.1	Banco de dados	45
4.2	Diagrama de Persistência	47
4.3	Imagem de Persistência	52
4.4	Distância entre os Diagramas de Persistência	56
4.4.1	Distância Bottleneck	56
4.5	Distância entre as Imagens	63
4.5.1	Distância Euclidiana	69
4.5.2	Distância do Máximo	74
4.5.3	Distância da Soma	78
5	Resultados	83
5.1	MDS - <i>Multidimensional Scaling</i>	83
5.2	Dendrograma	88
5.2.1	<i>K</i> -means	97
6	Conclusão	103
	Referências	105
A	Lista de bibliotecas utilizadas na pesquisa	107
A.1	Documentação das Bibliotecas	108
B	QR-Codes	109
B.1	Notas de Rodapé	109

B.1.1	Agradecimentos	109
B.1.2	Capítulo 1	109
B.1.3	Capítulo 2	110
B.1.4	Capítulo 3	110
B.1.5	Capítulo 4	110
B.2	Referências Bibliográficas	111
B.3	Site	111

1 Introdução

A Análise Topológica de Dados (Topological Data Analysis, ou TDA, em inglês) é um ramo recente da Matemática que tem se desenvolvido de maneira bastante rápida e obtido sucessos em aplicações, como, por exemplo, na identificação de subtipos de câncer de mama, na classificação de objetos tridimensionais, na classificação de lesões hepáticas, na identificação de dois tipos de diabetes, entre outras. [18]

O trabalho será desenvolvido com o objetivo de agrupar (cluster) imagens tridimensionais. Para isso, no Capítulo 2, é feita uma introdução teórica às ferramentas topológicas: simplexes, complexos, triangularização, ciclo, bordo e grupos de homologia.

No Capítulo 3 as definições envolvem a base para a análise topológica de dados. Informações sobre tipos de complexos, filtrações e persistência são desenvolvidas. O Diagrama de persistência e Barcode são apresentados para as primeiras análises. Neste trecho, são também desenvolvidos, os conceitos de Distância entre diagramas de persistência e imagem de persistência, estes, essenciais para o desenvolvimento do estudo.

É no Capítulo 4 que o banco de dados das imagens tridimensionais é apresentado e lapidado.

Por fim, o Capítulo 5 avalia os resultados do estudo por meio do Escalonamento Multidimensional (MDS), do Dendrograma e pelo método de clusterização K -means.

Todas as funções elaboradas para o estudo foram desenvolvidas em linguagem de programação `Python` e estão exibidas no texto. No apêndice A está a biblioteca completa utilizada para a execução dos programas, bem como o link para algumas informações sobre elas.

O apêndice B traz como facilidade para o leitor tradicional, àquele que prefere o toque, os códigos QR's de todas as referências do trabalho (bibliográficas e notas de rodapé), além do acesso à um site, onde estão disponíveis todas as imagens utilizadas no estudo e alguns gráficos de difícil leitura.

2 Conhecimento preliminar

A topologia é uma abstração da geometria; trata de conjuntos tendo uma estrutura que permite a definição de continuidade para funções e um conceito de “proximidade” de pontos e conjuntos. Essa estrutura, chamada de “topologia” no conjunto, foi originalmente determinada a partir das propriedades de conjuntos abertos em espaços euclidianos, particularmente no plano euclidiano. (PICCORILLO, Lisa).¹

Para que o desenvolvimento do trabalho faça sentido e possa ser compreendido, algumas definições e resultados são essenciais. Este capítulo tem como referência o livro do Croom (1978) [9], a dissertação de Mestrado do Ronchi (2020) [20], o artigo de Cerda (et al., 2021) [3], as notas de aula do Menegon (2021) [15] e o livro do Munkres (1930) [16].

Definição 2.1 (Pontos geometricamente independentes). Seja $A = \{a_0, a_1, \dots, a_k\}$ um conjunto de $k+1$ pontos em $\mathbb{R}^n$ é dito **geometricamente independente** quando nenhum hiperplano² de dimensão $k-1$ contém todos os pontos de A .

Assim, o conjunto $\{a_0, a_1, \dots, a_k\}$ ser geometricamente independente, significa que todos os pontos são distintos; que quaisquer três pontos não pertencem à uma reta, quaisquer quatro pontos não pertencem à um plano, e, em geral, quaisquer $p+1$ pontos não pertencem à um hiperplano de dimensão $p-1$ ou menos.

Observação 2.2. Equivalentemente, os pontos $a_0, \dots, a_n$ são tais que os vetores diferença $a_1 - a_0, a_2 - a_0, \dots, a_n - a_0$ são linearmente independentes em $\mathbb{R}^m$.

Exemplo 2.3. Na Figura 2.1, os pontos $\{A, B, C\}$ são geometricamente dependentes, pois pertencem a um hiperplano de $\mathbb{R}^2$, ou seja, uma reta. Já os pontos $\{A, B, D\}$, $\{A, C, D\}$ ou $\{B, C, D\}$ são geometricamente independentes.

¹Disponível em: <https://www.bbc.com/portuguese/geral-53020787>.

²Disponível em: <https://eu-provo.blogspot.com/2022/02/dissertacao-mestrado.html>.

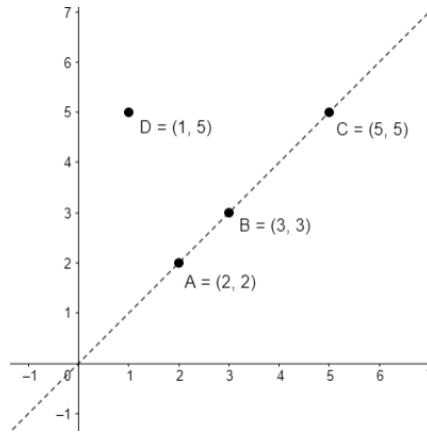


Figura 2.1: Pontos Geometricamente dependentes e independentes.

Fonte: Própria autora

Definição 2.4 (k -simplexo). Seja $\{a_0, \dots, a_k\}$ um conjunto de $k + 1$ pontos geometricamente independentes no $\mathbb{R}^n$. O **simplexo geométrico k -dimensional, ou k -simplexo**, σ^k , gerado por estes pontos é o conjunto de todos os pontos x em $\mathbb{R}^n$ para o qual existem números reais não negativos $\lambda_0, \dots, \lambda_k$ de modo que

$$x = \sum_{i=0}^k \lambda_i a_i \quad \text{e} \quad \sum_{i=0}^k \lambda_i = 1$$

Observe a Figura 2.2. Os números $\lambda_0, \dots, \lambda_k$ são as **coordenadas baricênticas**³ do ponto x e os pontos $a_0, \dots, a_k$ são os **vértices** de σ^k .

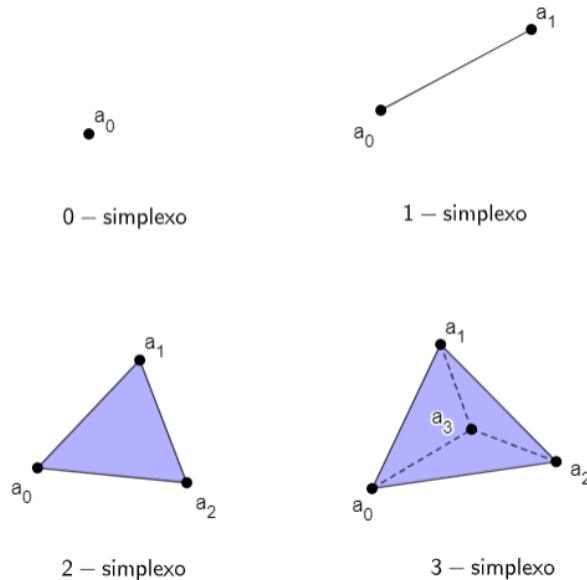


Figura 2.2: Exemplos de k -simplexos, com $k \in \{0, 1, 2, 3\}$.

Fonte: Própria autora.

³Disponível em: <https://eu-provo.blogspot.com/2022/04/coordenadas-baricentricas.html>.

Definição 2.5 (k -simplexo aberto). O conjunto de todos os pontos x , com todas as coordenadas baricêntricas positivas, do simplexo σ^k são chamados de **k -simplexo geométrico aberto** gerado por $\{a_0, \dots, a_k\}$.

Também conhecido como **interior** de σ^k , é definido por $\text{Int}(\sigma^k) = \sigma^k - \partial\sigma^k$, onde $\partial\sigma^k$ é a fronteira (2.9) de σ^k .

Observe na imagem 2.2 que o 0-simplexo é um conjunto unitário, um 1-simplexo é uma **aresta**, um 2-simplexo é um **triângulo** (interior e bordo), e um 3-simplexo é um **tetraedro** (interior e bordo). Um 0-simplexo aberto é, também, um conjunto unitário, um 1-simplexo aberto é um segmento sem as extremidades, um 2-simplexo aberto é o interior de um triângulo, e um 3-simplexo aberto é o interior do tetraedro.

Definição 2.6 (Dimensão de um simplexo). Seja σ^k um k -simplexo gerado pelos vértices $\{a_0, \dots, a_k\}$. O número k é a **dimensão do simplexo** σ .

Definição 2.7 (Faces de um simplexo). Qualquer simplexo gerado por um subconjunto de $\{a_0, \dots, a_k\}$ é uma **face** de σ^k ; assim, dizer que um simplexo σ^m é uma face de um simplexo σ^k , $m \leq k$, significa que cada vértice de σ^m é um vértice de σ^k . Por outro lado, temos que σ^k é **coface** de σ^m .

As faces de σ^k geradas por $\{a_1, \dots, a_k\}$ são chamadas **faces opostas** à a_0 .

As faces de σ^k , com exceção da própria σ^k , são chamadas **faces próprias** de σ^k .

O simplexo σ^k gerado por $\{a_0, \dots, a_k\}$ será denotado por $\sigma^k = \langle a_0, \dots, a_k \rangle$.

Exemplo 2.8. Seja o 2-simplexo $\sigma^2 = \langle a_0, a_1, a_2 \rangle$. Acompanhe na Figura 2.3. A dimensão de σ^2 é 2, suas faces são o próprio 2-simplexo, juntamente com os 1-simplexos $\langle a_0, a_1 \rangle$, $\langle a_1, a_2 \rangle$ e $\langle a_0, a_2 \rangle$, e os 0-simplexos $\langle a_0 \rangle$, $\langle a_1 \rangle$ e $\langle a_2 \rangle$. Assim, $\langle a_0, a_1 \rangle$ é uma 1-face de σ^2 e σ^2 é uma 2-coface de $\langle a_0, a_1 \rangle$. As **faces próprias** são os 1-simplexos $\langle a_0, a_1 \rangle$, $\langle a_1, a_2 \rangle$ e $\langle a_0, a_2 \rangle$, e os 0-simplexos $\langle a_0 \rangle$, $\langle a_1 \rangle$ e $\langle a_2 \rangle$. As **faces opostas** à $\langle a_0 \rangle$ são o 1-simplexo $\langle a_1, a_2 \rangle$ e os 0-simplexos $\langle a_1 \rangle$ e $\langle a_2 \rangle$.



Figura 2.3: Tipos de face do σ^2 .

Fonte: Própria autora.

Definição 2.9. A união de todas as faces próprias de um simplexo σ^k é a fronteira de σ^k , denotada por $\partial\sigma^k$.

Observação 2.10. Um q -simplexo tem $\binom{q+1}{d+1}$ d -faces e $\sum_{d=0}^q \binom{q+1}{d+1} = 2^{q+1}$ faces no total.

De fato, para $d = 0$, temos:

$$\binom{q+1}{0+1} = \binom{q+1}{1} = \frac{(q+1)!}{1!(q+1-1)!} = \frac{(q+1)q!}{1!q!} = q+1$$

que equivale a quantidade de vértices de um simplexo.

Para o total de faces:

$$\sum_{d=0}^q \binom{q+1}{d+1} = \binom{q+1}{1} + \binom{q+1}{2} + \cdots + \binom{q+1}{q+1} = (q+1) + \cdots + 1 = 2^{q+1}$$

pelo Teorema das linhas do Triângulo de Pascal.⁴

Exemplo 2.11. Seja $\sigma^2 = \langle a_0, a_1, a_2 \rangle$ o 2-simplexo do exemplo 2.8. Verifiquemos a quantidade de faces de σ^2 .

Pela observação 2.10, temos que:

$$\sum_{d=0}^2 \binom{2+1}{d+1} = \binom{3}{1} + \binom{3}{2} + \binom{3}{3} = \frac{3!}{1!2!} + \frac{3!}{2!1!} + 1 = 3 + 3 + 1 = 7$$

Definição 2.12 (Simplexos propriamente unidos). Dois simplexos σ^m e σ^n são **propriamente unidos** se eles não possuem interseção ou se $\sigma^m \cap \sigma^n$ for uma face de ambos. Observe exemplos na Figura 2.4.

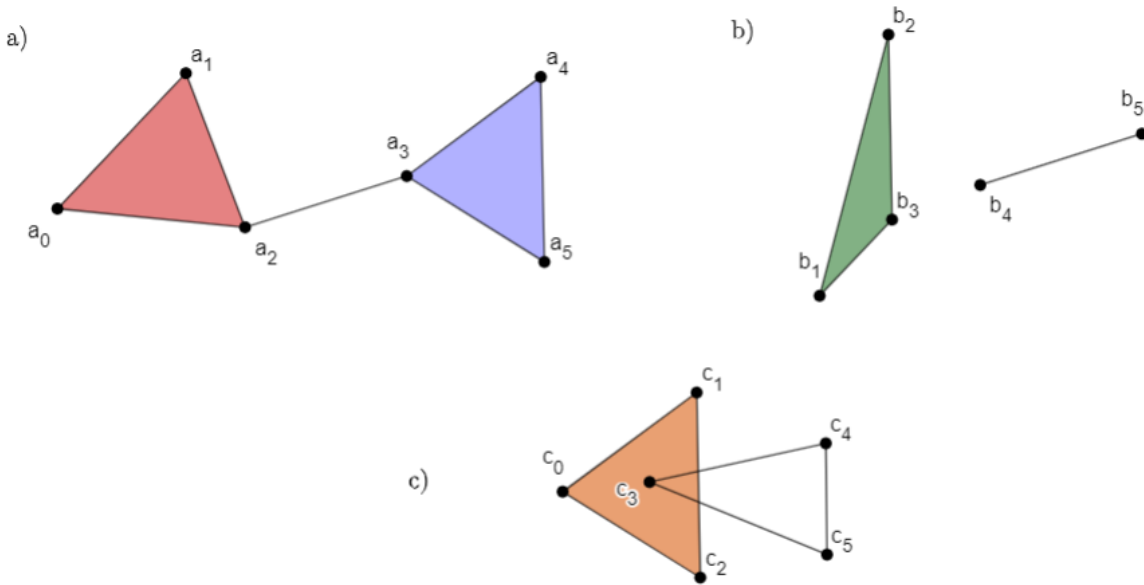


Figura 2.4: Os simplexos em a) e b) são propriamente unidos, os do item c) é uma união imprópria.

Fonte: Própria autora.

Definição 2.13 (Complexo simplicial geométrico). Um **complexo geométrico** (ou **complexo simplicial** ou **complexo**) é uma família finita K de simplexos geométricos unidos propriamente e que apresentam uma propriedade onde, cada face de um elemento de K é também um elemento de K , ou seja, dado $\sigma \in K$, temos que para toda face $\tau \subset \sigma$, satisfaz $\tau \in K$.

A **dimensão** de K é o maior inteiro positivo m tal que K tenha um m -simplexo.

⁴Disponível em: [https://eu-provo.blogspot.com/2022/04/teorema\[...\].html](https://eu-provo.blogspot.com/2022/04/teorema[...].html)

Definição 2.14 (Complexo simplicial abstrato). Seja X um conjunto finito com pontos quaisquer. Seja Y um conjunto de subconjuntos não-vazios de X . Dizemos que Y é um **complexo simplicial abstrato** de X se para todo $\sigma \in Y$, temos que todo subconjunto $\sigma' \subset \sigma$, $\sigma' \in Y$.

Cada elemento $\sigma \in Y$ é chamado de **simplexo**.

Exemplo 2.15. Sejam $X = \{a, b, c\}$ e $Y = \{\{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}\}$. Temos que Y é um complexo simplicial abstrato. Tome o simplexo $\sigma = \{a, b\}$. Note que seus subconjuntos $\{a\}$ e $\{b\}$ pertencem a Y . O mesmo acontece para o simplexo $\tau = \{a, c\}$. Logo, Y é, de fato, um complexo simplicial abstrato.

Definição 2.16 (Esquema de vértices). Sejam K é um complexo simplicial e V o conjunto dos vértices de K . Tome como $\mathcal{K}$ a coleção de todos os subconjuntos $\{a_0, \dots, a_n\}$ de V tal que os vértices $\{a_0, \dots, a_n\}$ gerem um simplexo de K . A coleção $\mathcal{K}$ é chamada de **esquema de vértices** de K .

Observação 2.17. A coleção $\mathcal{K}$ é um exemplo particular de um complexo simplicial abstrato.

Teorema 2.18.

- a) *Todo complexo abstrato $\mathcal{S}$ é isomorfo ao esquema de vértices de algum complexo simplicial K .*
- b) *Dois complexos simpliciais são linearmente isomorfos se, e somente se, seus esquemas de vértices são isomorfos como complexos simpliciais abstratos.*

Demonstração.

- a) Dado um conjunto de índices J , seja Δ^J a coleção de todos os simplexos E^J gerados por subconjuntos finitos de uma base $\{\epsilon_\alpha\}$ de E^J . Assim, Δ^J é um complexo simplicial.

Se σ e τ são simplexos de Δ^J , então a união do conjunto de seus vértices é geometricamente independente e gera um simplexo de Δ^J . Δ^J será chamado por **complexo de dimensão infinita**.

Agora, considere um complexo abstrato $\mathcal{S}$ com um conjunto de vértices V . Escolha um conjunto de índices J grande o suficiente para que haja uma função injetora $f: V \rightarrow \{\epsilon_\alpha\}_{\alpha \in J}$ (pode ser considerado $J = V$).

Um subcomplexo K de Δ^J é o complexo geométrico gerado por $f(a_0), \dots, f(a_n)$ de um simplexo abstrato $\{a_0, \dots, a_n\} \in \mathcal{S}$. Logo, K é um complexo simplicial e $\mathcal{S}$ é isomorfo ao esquema de vértices de K ; f é a correspondência necessária entre seus conjuntos de vértices.

- b) Segue imediatamente do Lema abaixo, disponível na página 22 da referência [16].

Lema: Seja $f: K \rightarrow L$ uma aplicação bijetora tal que os vértices $v_0, \dots, v_n \in K$ geram um simplexo em K se, e somente se, $f(v_0), \dots, f(v_n)$ gerarem um simplexo em L . Então, a aplicação simplicial induzida $g: |K| \rightarrow |L|$ é um homeomorfismo (bijetora com inversa contínua). A aplicação g é chamada de **homeomorfismo simplicial** ou **isomorfismo**, de K com L .

Demonstração Lema: Cada simplexo $\sigma \in K$ é levado por g à um simplexo $\tau \in L$, de mesma dimensão.

Para mostrar o homeomorfismo, é suficiente mostrar que $h: \tau \rightarrow \sigma$ induzida pela correspondência de vértices f^{-1} é a inversa de $g: \sigma \rightarrow \tau$.

Note que se $x = \sum t_i v_i$, então $g(x) = \sum t_i f(v_i)$, por definição, assim,

$$h(g(x)) = h(\sum t_i f(v_i)) = \sum t_i f^{-1}(f(v_i)) \stackrel{\text{bijetora}}{=} \sum t_i v_i = x. \quad \square$$

Definição 2.19 (Realização geométrica). Se o complexo simplicial abstrato $\mathcal{S}$ é isomorfo com o esquema de vértices do complexo simplicial K , chamamos K de **realização geométrica** de $\mathcal{S}$. Ela é unicamente determinada por um isomorfismo linear.

Exemplo 2.20. Considere o complexo abstrato $Y = \{\{a_0, a_1, a_2, a_3\}, \{a_0, a_1, a_2\}, \{a_0, a_1, a_3\}, \{a_0, a_2, a_3\}, \{a_1, a_2, a_3\}, \{a_0, a_1\}, \{a_1, a_2\}, \{a_2, a_0\}, \{a_0, a_3\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_0\}, \{a_1\}, \{a_2\}, \{a_3\}\}$. A realização geométrica pode ser feita de mais de uma forma, Figura 2.5:

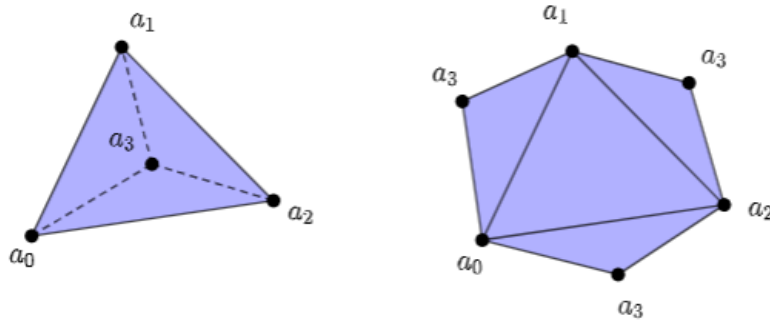


Figura 2.5: Realização geométrica.

Fonte: Própria autora.

Definição 2.21 (Operador geométrico). A união dos elementos de K com o subespaço topológico Euclidiano é denotado por $|K|$ e é chamado de **operador geométrico** de K ou um **poliedro associado com K** .

Observação 2.22. Todo espaço métrico compacto pode ser indefinidamente aproximado por poliedros, garante o teorema de P. S. Alexandroff (1928). Assim, ao trabalharmos complexos com números finitos de simplexos, nos dá a garantia da existência.

Definição 2.23 (Espaço triangularizável). Seja X um espaço topológico. Se existe um complexo geométrico K cujo operador geométrico $|K|$ é homeomorfo a X , então X é dito ser um **espaço triangularizável**, e o complexo K é chamado de **triangularização** de X .

Definição 2.24 (k -simplexo orientado). Um **k -simplexo orientado**, $k \geq 1$, é obtido de um k -simplexo $\sigma^k = \langle a_0, \dots, a_k \rangle$ escolhendo uma ordem para seus vértices. A orientação é indicada como na Figura 2.6.

Observação 2.25. A ordem dos vértices de um simplexo realmente importa.

$$\langle v_0, v_1, \dots, v_n \rangle \neq \langle v_1, v_0, \dots, v_n \rangle$$

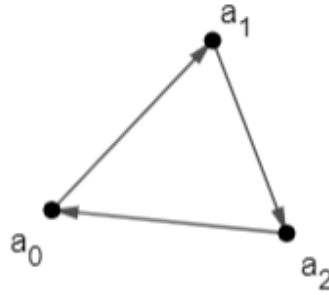


Figura 2.6: 2-simplexo orientado $\tau^2 = \langle a_0, a_1, a_2 \rangle$.

Fonte: Própria autora.

Definição 2.26 (Cadeia). Seja K um complexo simplicial. Uma p -cadeia em K é uma função c_p do conjunto de p -simplexos orientados σ^p de K para os inteiros, $c_p: K \rightarrow \mathbb{Z}$, tal que $c_p(-\sigma^p) = -c_p(\sigma^p)$, onde $-\sigma^p$ é a orientação oposta de σ^p .

A **0-cadeia** é uma função dos 0-simplexos de K para os números inteiros.

Com a operação de adição pontual induzida pelos inteiros, a família de p -cadeias forma um grupo chamado **grupo de p -cadeia de K** . Este grupo é denotado por $C_p(K)$. Se $p < 0$ ou $p > \dim K$, $C_p(K)$ representa o grupo trivial.

Se σ^p é um p -simplexo orientado, a **cadeia elementar** c_p correspondente a σ^p é a função definida da seguinte forma:

- $c_p(\sigma^p) = 1$,
- $c_p(-\sigma^p) = -1$,
- $c_p(\tau^p) = 0$, para cada p -simplexo orientado $\tau^p \neq \sigma^p$.

Essa p -cadeia elementar é denotada por $g\sigma^p$, onde $g = c_p(\sigma^p) \in \mathbb{Z}$. Com essa notação, uma p -cadeia d_p arbitrária pode ser expressa como uma soma formal finita,

$$d_p = \sum_i g_i \sigma_i^p, \quad g_i \in \mathbb{Z}$$

de p -cadeias elementares, onde o índice i varia em todos os p -simplexos de K .

Observação 2.27. Por abuso de notação, usamos o símbolo σ para denotar não apenas um simplexo, ou um simplexo orientado, mas também para denotar a cadeia elementar c correspondente ao simplexo orientado σ . Para facilitar a distinção, sendo $\sigma^k = \langle a_0, \dots, a_k \rangle$ um simplexo orientado, a k -cadeia correspondente será denotada por $[a_0, \dots, a_k]$.

Observação 2.28. O grupo de cadeias $C_p(K)$ é isomorfo à soma direta do grupo $\mathbb{Z}$ (ou qualquer outro grupo ou anel comutativo) sobre a família de p -simplexos de K . Isto é, se K tem α_p p -simplexos, então $C_p(K)$ é isomorfo à soma direta de α_p cópias de $\mathbb{Z}$. Um isomorfismo é dado pela correspondência

$$\sum_{i=1}^{\alpha_p} g_i \sigma_i^p \Leftrightarrow (g_1, g_2, \dots, g_{\alpha_p}) \Leftrightarrow \mathbb{Z}_1 \oplus \mathbb{Z}_2 \oplus \dots \oplus \mathbb{Z}_{\alpha_p}$$

Definição 2.29 (Homomorfismo fronteira). Seja $\sigma^p = \langle a_0, \dots, a_p \rangle$ um p -simplexo orientado de um K complexo qualquer, com $p > 0$. Definimos o homomorfismo fronteira $\partial_p: C_p(K) \rightarrow C_{p-1}(K)$, tal que:

$$\partial_p(\sigma^p) = \partial_p[a_0, \dots, a_p] = \sum_{i=0}^n (-1)^i [a_0, \dots, \hat{a}_i, \dots, a_n],$$

onde $\hat{a}_i$ significa que o vértice a_i foi removido da sequência.

Como $C_p(K)$ é o grupo trivial para $p < 0$, o operador ∂_p é o homomorfismo trivial para $p \leq 0$.

Exemplo 2.30. Calcular as fronteiras dos simplexos:

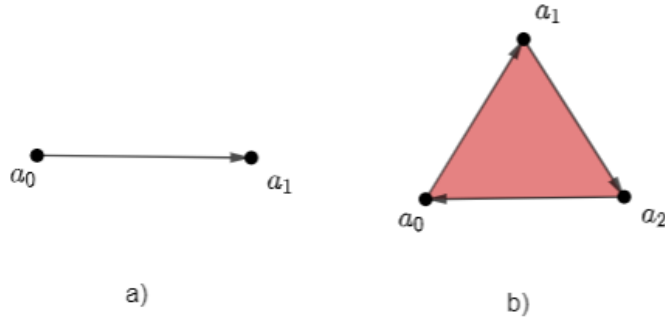


Figura 2.7: a) 1-simplexo $[a_0, a_1]$, b) 2-simplexo $[a_0, a_1, a_2]$.

$$\text{a) } \partial_1[a_0, a_1] = (-1)^0[a_1] + (-1)^1[a_0] = [a_1] - [a_0]$$

$$\text{b) } \partial_2[a_0, a_1, a_2] = (-1)^0[a_1, a_2] + (-1)^1[a_0, a_2] + (-1)^2[a_0, a_1] = [a_1, a_2] - [a_0, a_2] + [a_0, a_1]$$

Definição 2.31 (Ciclo). Seja K um complexo orientado. Se p é um inteiro positivo, um p -ciclo em K é uma p -cadeia z_p tal que $\partial_p(z_p) = 0$.

O kernel (ou núcleo) do $\partial_p: C_p(K) \rightarrow C_{p-1}(K)$ é um subgrupo de $C_p(K)$ chamado de **grupo de p -ciclos** e denotado por $Z_p(K)$ (por conta da palavra alemã “Zyklus” que significa “ciclo”).

Definição 2.32 (Bordo). Se $p \geq 0$, a p -cadeia b_p é um p -bordo em K , se houver $(p+1)$ -cadeia σ_{p+1} tal que $\partial_{p+1}(\sigma_{p+1}) = b_p$.

A imagem do $\partial_{p+1}: C_{p+1}(K) \rightarrow C_p(K)$ é um subgrupo de $C_p(K)$ chamada de **grupo de p -bordos** e denotado por $B_p(K)$.

Observação 2.33.

- O 0-ciclo, tal como a 0-cadeia, é zero; logo, o grupo $Z_0(K)$ é trivial.
- Se n é a dimensão de K , então não há p -cadeia em K para $p > n$.
- Intuitivamente, um p -ciclo é uma combinação linear de p -simplexos que faz um circuito completo.
- Os p -ciclos que circundam “buracos” são aqueles que não são bordos, isto é, $Z_p(K) \not\subset B_p(K)$.

- A composição⁵ $\partial_n \circ \partial_{n+1} = 0$, ou seja, $B_p(K) \subset Z_p(K)$.

Definição 2.34 (Grupo de Homologia). Se K é um complexo orientado e p é um inteiro não negativo, o p -ésimo grupo de homologia de K é dado pelo quociente:

$$H_p(K) = Z_p(K)/B_p(K)$$

e as classes $[z_p] \in H_p(K)$ são chamadas de **classes de homologia**.

Observação 2.35. Dois ciclos representando a mesma classe de homologia são ditos homólogos. Isso significa que a diferença entre eles é uma fronteira.

Definição 2.36 (Complexo de cadeias). Uma sequência de homomorfismos de grupos abelianos

$$C_{n+1} \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \rightarrow \cdots \rightarrow C_1 \xrightarrow{\partial_1} C_0 \xrightarrow{\partial_0} 0$$

com $\partial_n \circ \partial_{n+1} = 0$ para cada n , é dita um **complexo de cadeias**.

⁵Disponível na aula 15, pagina 6 da referência [15] e na página 14 da referência [23].

3 Diagrama de Persistência

3.1 Complexos

Como visto no capítulo anterior, um complexo simplicial é uma família de simplexes (2.13) e eles são a base para a construção dos resultados posteriores, como ciclos (2.31), bordos (2.32) e os grupos de homologia (2.34). Foi visto, que a realização geométrica (2.19) nem sempre é única, assim, para um conjunto de pontos, é possível construir os complexos segundo alguns parâmetros. Veremos as opções clássicas.¹

3.1.1 Complexo de Čech

Definição 3.1 (Complexo de Čech). Sejam X um conjunto de pontos em $\mathbb{R}^n$ e $r > 0$ um valor fixo real. O **complexo de Čech** é um complexo simplicial onde os vértices (0-simplexos) são elementos de X e o conjunto de pontos $\{x_0, \dots, x_k\}$ define um k -simplexo se

$$\bigcap_{i=0}^k B(x_i, r) \neq \emptyset,$$

onde $B(x_i, r) = \{x \in \mathbb{R}^n \mid d(x_i, x) < r\}$ é uma bola aberta centrada em x_i , com raio r , para uma distância d . Veja na Figura 3.1.

Denotamos o complexo de Čech para um valor real $r > 0$ de um conjunto X por $C^r(X)$.

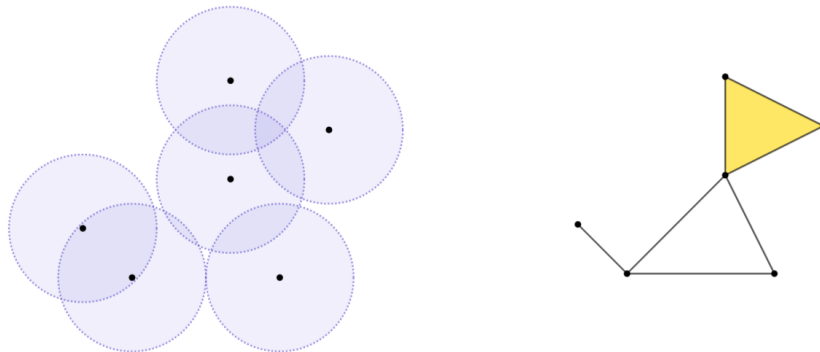


Figura 3.1: Dado um conjunto de pontos, à esquerda temos um parâmetro r , com a distância euclidiana aplicado e à direita, o Complexo de Čech resultante.

Fonte: Própria autora.

¹Nesta seção as definições e resultados tem como referência a dissertação [20] e o artigo [3].

A ideia do Complexo de Čech é que, se duas bolas se intersectam, adicionamos um 1-simplexo (aresta), se três bolas se intersectam, adicionamos um 2-simplexo, ou seja, um k -simplexo é adicionado ao complexo simplicial abstrato quando há interseção de $k + 1$ bolas, centradas nos vértices, com raio $r > 0$ e distância d .

3.1.2 Complexo de Vietoris-Rips (VR)

Definição 3.2 (VR Complexo). Sejam X um conjunto de pontos em $\mathbb{R}^n$ e $r > 0$ um valor fixo real. O **complexo de VR** é um complexo simplicial onde os vértices (0-simplexos) são elementos de X e o conjunto de pontos $\{x_0, \dots, x_k\}$ define um k -simplexo, $\sigma^k = \langle x_0, \dots, x_k \rangle$, se $\text{diam}(\sigma^k) \leq 2r$. Considere a Figura 3.2.

Denotamos o VR complexo para um valor real r de um conjunto X por $\text{VR}^r(X)$.

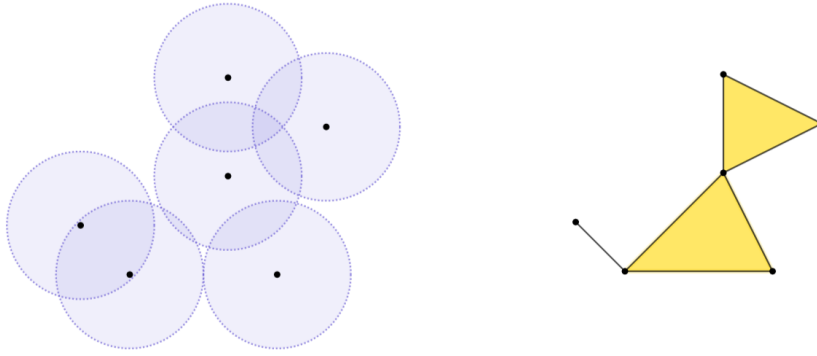


Figura 3.2: Dado um conjunto de pontos, à esquerda temos um parâmetro r , com a distância euclidiana aplicado e à direita, o VR complexo resultante.

Fonte: Própria autora.

Para o Complexo de Vietoris-Rips, um k -simplexo é adicionado ao complexo simplicial abstrato se a distância entre os vértices, dois à dois, for menor ou igual a $2r$, onde r é o raio das bolas centradas nos vértices, construídas por uma distância d .

Observação 3.3. Temos que $C^r(X) \subseteq \text{VR}^r(X) \subseteq C^{2r}(X)$ como relação entre os complexos.

Demonstração.

Seja $X \subset \mathbb{R}^n$ um conjunto de pontos.

Tome $\sigma^k = \langle x_0, \dots, x_k \rangle$ um k -simplexo, tal que $x_0, \dots, x_k \in X$.

Se $\sigma^k \in C^r(X)$, então $\bigcap_{i=0}^k B(x_i, r) \neq \emptyset$.

Como $\bigcap_{i=0}^k B(x_i, r) \neq \emptyset$, temos que $\text{diam}(\sigma^k) \leq 2r$.

Logo, $\sigma^k \in \text{VR}^r(X)$.

Da mesma forma, se $\text{diam}(\sigma^k) \leq 2r$, então $\bigcap_{i=0}^k B(x_i, 2r) \neq \emptyset$.

Portanto, $C^r(X) \subseteq \text{VR}^r(X) \subseteq C^{2r}(X)$. $\square$

Observação 3.4. Do ponto de vista computacional, o complexo VR é mais viável (ou seja, menor armazenamento e complexidade de tempo) do que Čech. Pois, no complexo de Čech o computador precisa armazenar todas as distâncias e verificar se há interseção em cada nível; já no complexo de VR, a informação sobre as distâncias são suficientes para a construção dos simplexos.

3.1.3 Complexo Alpha

Definição 3.5 (Diagrama de Voronoi). Dado um subconjunto $X = \{x_0, \dots, x_k\}$, $X \subset \mathbb{R}^n$, definimos a **célula de Voronoi** associada ao ponto x_i sendo:

$$V_i = \{x \in \mathbb{R}^n \mid d(x_i, x) \leq d(x_j, x), \forall j \in \{0, \dots, k\}\},$$

em que d é a distância euclidiana usual. Como mostra a Figura 3.3.

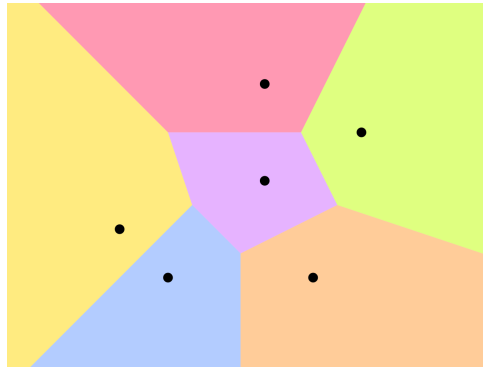


Figura 3.3: Diagrama de Voronoi de um conjunto de pontos.

Fonte: Própria autora.

Na prática, para a construção do Diagrama, basta considerar as delimitações das mediatrizes dos pontos, dois à dois.

Definição 3.6 (Complexo Alpha). Sejam um conjunto de pontos X em $\mathbb{R}^n$ e $r > 0$ um valor fixo real. O **complexo Alpha** de X é um complexo simplicial onde os vértices (0-simplexos) são os elementos de X e o conjunto de pontos $\{x_0, \dots, x_k\}$ define um k -simplexo se,

$$\bigcap_{i=0}^k R(x_i, r) \neq \emptyset,$$

onde $R(x_i, r) = B(x_i, r) \cap V_i$, para todo $i \in \{0, \dots, k\}$. Observe a Figura 3.4.

Denotamos o complexo Alpha para um valor real r de um conjunto X por $A^r(X)$.

Observação 3.7. Para o complexo Alpha, temos a seguinte relação: $A^r(X) \subset C^r(X)$.

Demonstração.

Seja $X \subset \mathbb{R}^n$ um conjunto de pontos.

Tome $\sigma^k = \langle x_0, \dots, x_k \rangle$ um k -simplexo, tal que $x_0, \dots, x_k \in X$.

Se $\sigma^k \in A^r(X)$, então $\bigcap_{i=0}^k R(x_i, r) \neq \emptyset$.

Como $\bigcap_{i=0}^k R(x_i, r) = \bigcap_{i=0}^k (B(x_i, r) \cap V_i) \neq \emptyset$, temos que $\bigcap_{i=0}^k B(x_i, r) \neq \emptyset$.

Logo, $\sigma^k \in C^r(X)$.

Portanto, $A^r(X) \subset C^r(X)$. □

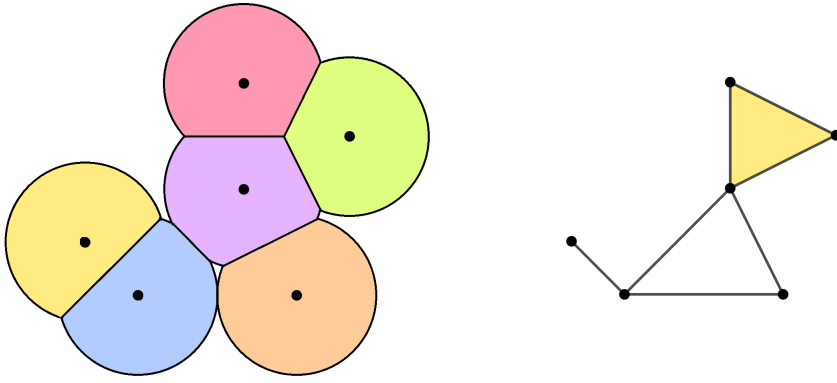


Figura 3.4: Dado um conjunto de pontos, à esquerda temos um parâmetro r interseccionado com Diagrama de Voronoi e à direita, o Complexo Alpha resultante.

Fonte: Própria autora.

3.1.4 Filtração

A ideia da filtração de um complexo, é separá-lo em subconjuntos de forma crescente, onde o seu menor subconjunto é composto pelos vértices do complexo e aumenta a medida que se anexa os simplexos de acordo com um parâmetro, até que atinja sua totalidade.

Para as definições que seguem, são mencionados o artigo [3], a dissertação [20] e o livro [11].

Definição 3.8 (Subcomplexo). Seja K um complexo simplicial. K' é um **subcomplexo** de K se $K' \subseteq K$ e K' for também um complexo simplicial.

Definição 3.9 (Filtração). Seja K um complexo simplicial. Uma **filtração** de K é uma sequência de subcomplexos $K_i \subseteq K$, de tal forma que,

$$\emptyset \subseteq K_0 \subseteq K_1 \subseteq \dots \subseteq K_{n-1} \subseteq K_n = K.$$

Em outras palavras, a filtração de um complexo é a sucessão de acréscimos de subcomplexos de K . A Figura 3.5 exemplifica.

Neste caso, K é dito **complexo simplicial filtrado**.

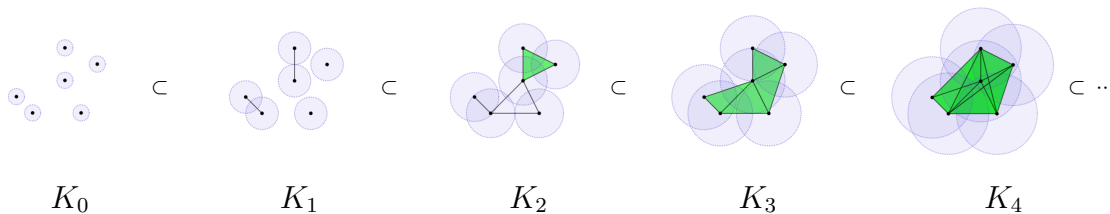


Figura 3.5: Fragmento de uma filtração de um complexo simplicial de VR obtida pela variação do parâmetro $r_i = (i + 1) \cdot 0,5$ para cada $K_i \in K$. Imagem fora de escala.

Fonte: Própria autora.

Quando os simplexos são determinados por proximidade de acordo com uma função distância, como nos casos do complexo de Čech, VR complexo e complexo Alpha, uma

filtração de um complexo simplicial K é obtida por uma sequência de valores positivos $0 < r_1 < r_2 < \dots < r_n$, onde o complexo K_i corresponde ao parâmetro r_i .

Definição 3.10 (Função Nível de Filtração). Seja K um complexo simplicial finito filtrado, $\emptyset \subseteq K_0 \subseteq K_1 \subseteq \dots \subseteq K_n = K$. A **função de nível de filtração** ψ é definida em $\{r_0, r_1, \dots, r_n\}$ por $\psi(r_i) = K_i$.

Definição 3.11 (Nascimento ou morte). **Nascimento** é um conceito para descrever o nível de filtração quando uma nova característica topológica aparece. Similarmente, **morte** refere-se ao nível de filtração quando uma característica topológica desaparece. Então, um **intervalo persistente** (nascimento, morte) é uma “vida” de uma característica topológica dada.

Uma filtração pode ser entendida como um método para construir todo o complexo simplicial K a partir de uma ‘família’ de sub-complexos ordenados incrementalmente de acordo com alguns critérios, onde cada nível i corresponde ao ‘nascimento’ ou ‘morte’ de uma característica topológica. [...]. [3]

Observação 3.12. Na filtração, os simplexes se unem aos criados primeiro; assim, o simplexo que morre na união, recebe o nome do simplexo mais antigo do complexo. Esta regra é chamada de **regra mais antiga**.

3.2 Homologia Persistente

A Topologia de Persistência consiste no estudo de propriedades de espaços topológicos com filtrações. A Topologia de Persistência permite analisar os dados representados por uma função filtração e examinar como as propriedades de seus conjuntos de subníveis persistem quando se percorre a filtração.[14]

Em um complexo simplicial filtrado, há uma evolução topológica expressa pela sequência de grupos de homologia.

Nesta seção, foram utilizadas as referências [11], [22] e [4].

Definição 3.13 (Sequência de grupos de homologia de uma filtração). Seja um complexo simplicial K com uma filtração,

$$\emptyset \subset K_1 \subset K_2 \subset \dots \subset K_{n-1} \subset K_n = K.$$

Definimos uma aplicação inclusão $f: K_i \rightarrow K_j$, onde $i \leq j$. Assim, temos um homomorfismo induzido $f_p^{i,j}: H_p(K_i) \rightarrow H_p(K_j)$, para cada dimensão p .

Logo, a filtração corresponde a **sequência de grupos de homologia** conectada por homomorfismos, para cada dimensão p .

$$0 = H_p(K_0) \rightarrow H_p(K_1) \rightarrow \dots \rightarrow H_p(K_{n-1}) \rightarrow H_p(K_n) = H_p(K).$$

O p -ésimo grupo de homologia persistente é a imagem do homomorfismo induzido pela inclusão, $\text{Im} \left(f_p^{i,j} \right) = H_p^{i,j}$, para $0 \leq i \leq j \leq n$.

A medida que avançamos na filtração, de K_{i-1} à K_i , podemos ganhar novas classes de homologia e podemos perder algumas quando se tornam triviais ou se fundem. Coletamos as classes que nasceram nela ou as que nasceram antes, e morreram após a união com outro grupo. Quando uma classe morre, ela se funde à outra mais antiga, de acordo com a sequência dos simplexos criados.

Os grupos de homologia persistente consistem nas classes de homologia de K_i que ainda estão vivas em K_j ou, mais formalmente, $H_p^{i,j} = Z_p(K_i) / (B_p(K_j) \cap Z_p(K_i))$. Estes, são espaços vetoriais² com coeficientes em um corpo; neste trabalho, usaremos o corpo $\mathbb{Z}_2$. Temos tal grupo para cada dimensão p e cada par de índices $i \leq j$.

Definição 3.14 (Nascimento e morte na sequência de grupos de homologia). Seja γ uma classe de homologia de dimensão p em $H_p(K_i)$, isto é, $\gamma \in H_p(K_i)$. Dizemos que γ **nasce** em K_i se $\gamma \notin \text{Im}(f_p^{i-1,i})$; e, se γ nasce em K_i , diremos que γ **morre** ao entrar em K_j se γ se fundir com uma classe mais velha, conforme vamos de K_{j-1} para K_j . Assim, $f_p^{i,j-1}(\gamma) \notin \text{Im}(f_p^{i-1,j-1})$, mas $f_p^{i,j}(\gamma) \in \text{Im}(f_p^{i-1,j})$.

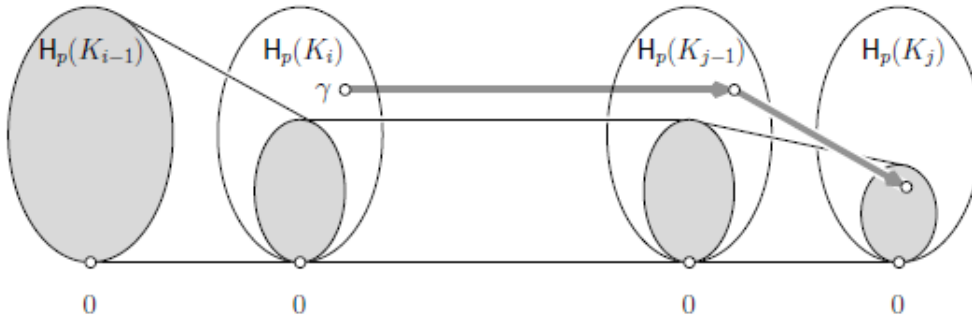


Figura 3.6: A classe γ nasce em K_i pois não está na imagem (sombreada) de $H_p(K_{i-1})$. Além disso, γ morre entrando em K_j já que esta é a primeira vez que sua imagem se funde com a imagem de $H_p(K_{i-1})$.

Fonte: (EDELSBRUNNER, HARER, 2010, p. 151)[11]

Definição 3.15 (Persistência). Se γ nasce em K_i e morre entrando em K_j , então chamamos a diferença no valor da função de **persistência**, $\text{pers}(\gamma) = a_j - a_i$ e representamos pelo par $(i, j) = (\text{nascimento}, \text{morte})$.

Às vezes preferimos ignorar os valores reais da função e considerar a diferença no índice, $j - i$, que chamamos de **persistência de índice da classe**. Se γ nasce em K_i mas nunca morre, então definimos sua persistência assim como sua persistência de índice para infinito, (i, ∞) .

Observação 3.16. Nascimento e mortes também podem ser definidos para uma sequência de espaços vetoriais que não são necessariamente grupos de homologia. Tudo o que se precisa é uma sequência finita e homomorfismos da esquerda para a direita, que, para espaços vetoriais, são geralmente chamados de funções lineares.

Existem duas formas de visualizar esses intervalos, através dos **barcodes** (código de barras) ou dos **diagramas de persistência** (PD). Essas ferramentas indicam em quais

²Disponível em: <https://eu-provo.blogspot.com/2022/07/espaco-vetorial.html>.

parâmetros de escala as características topológicas aparecem pela primeira vez (“nasce”) e não mais permanecem (“morre”). Em um barcode, um segmento de linha horizontal começa no nascimento de cada característica e termina quando ela morre, ou seja, desenharmos uma barra do comprimento do intervalo $[i, j)$. Por outro lado, um diagrama de persistência exhibe informações homológicas como pontos (i, j) no plano cartesiano. Os eixos horizontal e vertical codificam os parâmetros de nascimento e morte, respectivamente. Como todas as características morrem depois de nascer, significa que os pontos estarão no meio plano positivo, acima da diagonal. Normalmente, pontos próximos à diagonal são considerados como ruído, enquanto aqueles mais distantes da diagonal sugerem características topológicas mais significativas.

Exemplo 3.17. Considere o complexo simplicial de Čech da Figura 3.7, com uma filtração obtida pelo acréscimo do parâmetro $r_i = 0,5$ para K_i .

Tome a seguinte ordenação:

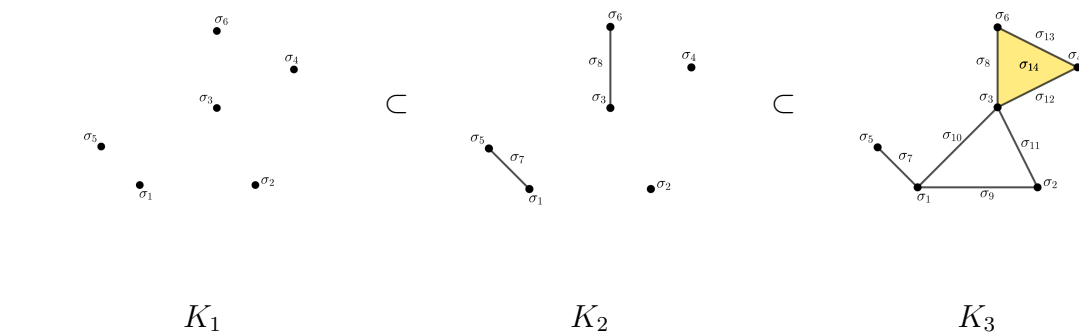
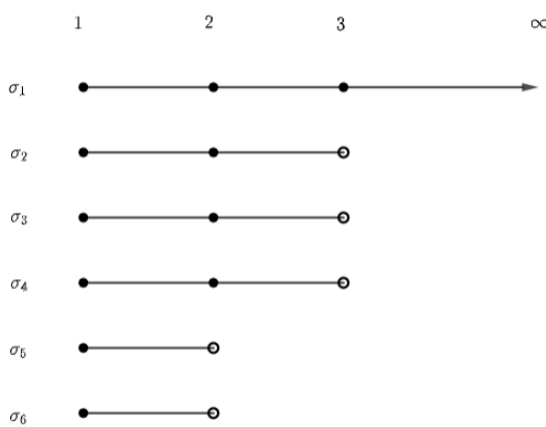


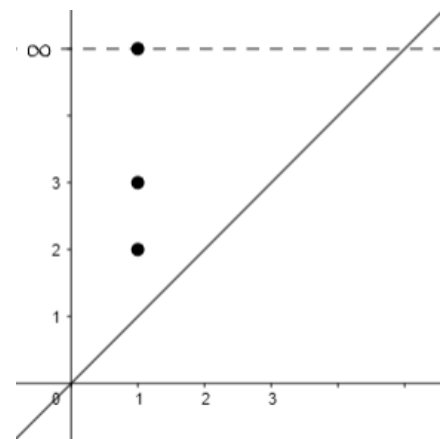
Figura 3.7: Complexo simplicial filtrado.

Fonte: Própria autora.

Para H_0 , temos:



(a) Barcode de H_0 .



(b) Diagrama de Persistência de H_0 .

Figura 3.8: Tipos de representação da persistência do complexo da Figura 3.7.

Fonte: Própria autora.

3.2.1 A matriz bordo ∂

Seja K um complexo simplicial filtrado e sejam $\sigma_1, \dots, \sigma_n$ os simplexes de K . Colocaremos uma ordenação de tal maneira que os simplexes de K_a precedam os simplexes de K_b , para $a < b$ e, se ambos estiverem no mesmo nível K_a da filtração, as faces dos simplexes são priorizadas, ou seja, se σ_i é face de σ_j , então $\sigma_i < \sigma_j$. Assim, a ordem de construção dos simplexes é preservada, bem como a construção de novos simplexes. Nesse contexto, apresentamos o seguinte:

Definição 3.18 (Matriz bordo). A **matriz bordo** de K , denotada por ∂ , é uma matriz $n \times n$ em que cada entrada tem o seguinte valor

$$\delta(i, j) = \begin{cases} 1, & \text{se o simplexo } \sigma_i \text{ é face de } \sigma_j \text{ e } \dim(\sigma_j) = \dim(\sigma_i) + 1 \\ 0, & \text{caso contrário.} \end{cases}$$

Na matriz bordo, cada coluna e cada linha representa um simplexo de K e, abusando da notação, citamos suas linhas e colunas com o nome dos simplexes.

Com a matriz construída, usaremos um algoritmo para sua redução, descrito a seguir.

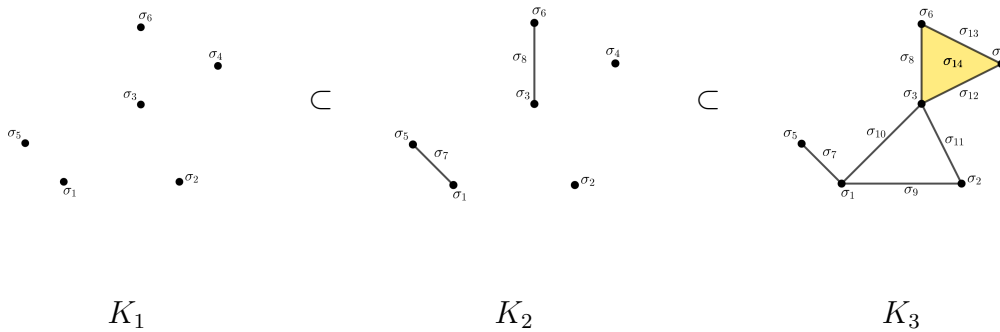
Para cada coluna σ_j , será denotado por $\text{low}(j)$ o maior inteiro $i \in \{1, \dots, n\}$ tal que $\delta(i, j) = 1$. Assim, para cada coluna σ_j , enquanto houver $\text{low}(j_d) = \text{low}(j_c)$, para $d > c$, com $c, d \in \{1, \dots, n\}$, faremos $\sigma_{j_d} = \sigma_{j_c} + \sigma_{j_d}$, ou seja, trocaremos uma coluna pela sua soma com outra coluna.

Definição 3.19 (Matriz reduzida). Uma matriz bordo estará **reduzida** quando, para quaisquer colunas j_c, j_d não nulas, $\text{low}(j_c) \neq \text{low}(j_d)$. Para as colunas nulas, o low é dito indefinido.

As colunas da matriz reduzida admitem as seguintes interpretações:

- quando a coluna de σ_j for nula, dizemos que o simplexo σ_j é **positivo**, pois nele nasce um ciclo;
- quando a coluna de σ_j for não-nula, existe um i tal que $\text{low}(j) = i$. Dizemos que o simplexo σ_j é **negativo**, pois nele morre um ciclo. Ciclo este, que nasceu com a adição com o simplexo σ_i .

Exemplo 3.20. Usaremos para a construção da matriz bordo a filtração do complexo simplicial da Figura 3.7. Encontraremos a matriz reduzida e faremos a sua interpretação.



$$\partial(K_2)_{14 \times 14} = \begin{matrix} & \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \sigma_7 & \sigma_8 & \sigma_9 & \sigma_{10} & \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \begin{matrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \\ \sigma_8 \\ \sigma_9 \\ \sigma_{10} \\ \sigma_{11} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{14} \end{matrix} & \left(\begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

Esta matriz é construída, usualmente, sobre o corpo $\mathbb{Z}_2 = \{0, 1\}$, assim, $1 + 1 = 0$. Analisando para reduzir a matriz, temos:

- $\text{low}(13) = \text{low}(8)$, logo $\sigma_{13} = \sigma_8 + \sigma_{13}$;
- $\text{low}(11) = \text{low}(10)$, logo $\sigma_{11} = \sigma_{10} + \sigma_{11}$;
- $\text{low}(11) = \text{low}(9)$, logo $\sigma_{11} = \sigma_9 + \sigma_{11}$;
- $\text{low}(13) = \text{low}(12)$, logo $\sigma_{13} = \sigma_{12} + \sigma_{13}$.

Após este processo, esta é a matriz resultante:

$$R(K_2) = \begin{matrix} & \sigma_1 & \sigma_2 & \sigma_3 & \sigma_4 & \sigma_5 & \sigma_6 & \sigma_7 & \sigma_8 & \sigma_9 & \sigma_{10} & \sigma_{11} & \sigma_{12} & \sigma_{13} & \sigma_{14} \\ \begin{matrix} \sigma_1 \\ \sigma_2 \\ \sigma_3 \\ \sigma_4 \\ \sigma_5 \\ \sigma_6 \\ \sigma_7 \\ \sigma_8 \\ \sigma_9 \\ \sigma_{10} \\ \sigma_{11} \\ \sigma_{12} \\ \sigma_{13} \\ \sigma_{14} \end{matrix} & \left(\begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

As colunas nulas $\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \sigma_{11}$ e σ_{13} são ditas positivas, pois é onde acontece o nascimento de ciclos: por meio dos 0-simplexos de σ_1 à σ_6 para H_0 ; dos 1-simplexos σ_{11} e σ_{13} para H_1 .

Fazendo um pareamento entre os simplexos, relacionando a coluna com o valor low correspondente, concluímos, pela definição de $\delta(i, j)$, que:

- $\text{low}(7) = 5 \Rightarrow \sigma_5$ é face de σ_7 e $\dim(\sigma_7) = \dim(\sigma_5) + 1$;
- $\text{low}(8) = 6 \Rightarrow \sigma_6$ é face de σ_8 e $\dim(\sigma_8) = \dim(\sigma_6) + 1$;
- $\text{low}(9) = 2 \Rightarrow \sigma_2$ é face de σ_9 e $\dim(\sigma_9) = \dim(\sigma_2) + 1$;
- $\text{low}(10) = 3 \Rightarrow \sigma_3$ é face de σ_{10} e $\dim(\sigma_{10}) = \dim(\sigma_3) + 1$;
- $\text{low}(12) = 4 \Rightarrow \sigma_4$ é face de σ_{12} e $\dim(\sigma_{12}) = \dim(\sigma_4) + 1$;
- $\text{low}(14) = 13 \Rightarrow \sigma_{13}$ é face de σ_{14} e $\dim(\sigma_{14}) = \dim(\sigma_{13}) + 1$;

Para cada relação, temos um intervalo de persistência, de acordo com o nível de nascimento do simplexo:

- $[\sigma_5, \sigma_7) = [1, 2)$, pois σ_5 nasceu em K_0 e σ_7 nasceu em K_1 ;
- $[\sigma_6, \sigma_8) = [1, 2)$, pois σ_6 nasceu em K_0 e σ_8 nasceu em K_1 ;
- $[\sigma_2, \sigma_9) = [1, 3)$, pois σ_2 nasceu em K_0 e σ_9 nasceu em K_2 ;
- $[\sigma_3, \sigma_{10}) = [1, 3)$, pois σ_3 nasceu em K_0 e σ_{10} nasceu em K_2 ;
- $[\sigma_4, \sigma_{12}) = [1, 3)$, pois σ_4 nasceu em K_0 e σ_{12} nasceu em K_2 ;
- $[\sigma_{13}, \sigma_{14}) = [3, 3)$, pois σ_{13} nasceu em K_2 e σ_{14} nasceu em K_2 ;
- $[\sigma_1, \infty) = [1, \infty)$, pois σ_1 nasceu em K_0 e não se relaciona com nenhum outro simplexo;
- $[\sigma_{11}, \infty) = [3, \infty)$, pois σ_{11} nasceu em K_2 e não se relaciona com nenhum outro simplexo.

Os simplexos que não se relacionam com nenhum outro são o σ_1 e o σ_{11} .

Portanto, o diagrama de persistência para H_0 da filtração é representado pelos pontos $(1, 2)$, $(1, 3)$ e $(1, \infty)$. É exatamente o mesmo obtido pela interpretação geométrica da filtração, na Figura 3.8b. Os pontos $(3, 3)$ e $(3, \infty)$ não são considerados pois pertencem à H_1 .

3.3 Distância Bottleneck

A capacidade de calcular distâncias entre diagramas de persistência permite concluir tarefas de Machine Learning (ML - aprendizado de máquina) que dependem apenas da distância entre objetos, como por exemplo, alguns algoritmos de agrupamento. Desta forma, os PDs podem ser pensados como uma estatística para agrupar os espaços topológicos subjacentes (ou outros objetos). [4]

A distância bottleneck, é uma métrica padrão utilizada para calcular a distância entre os Diagramas de Persistência.

Os resultados e definições apresentados nesta seção têm como referência as dissertações [20], [22] e o livro [11].

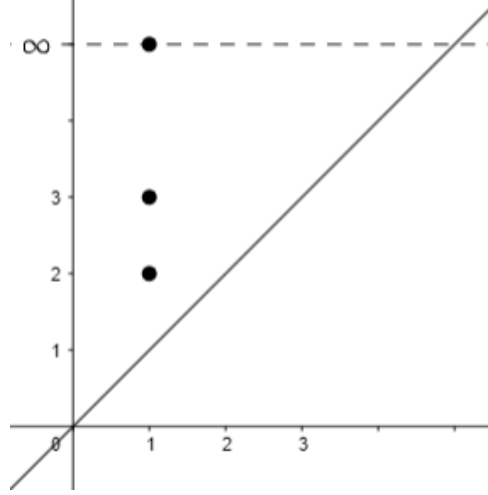


Figura 3.9: Diagrama de Persistência de H_0 do complexo simplicial filtrado
Fonte: Própria autora.

Definição 3.21 (Métrica L^∞). Sejam $x = (a, b)$ e $y = (p, q)$ dois pontos do plano. A métrica L^∞ é dada por:

$$d_\infty(x, y) = d_\infty((a, b), (p, q)) = \max\{|a - p|, |b - q|\}.$$

Para pontos no infinito, é dada da seguinte forma:

$$\begin{aligned} d_\infty((-\infty, b), (-\infty, q)) &= |b - q|, \\ d_\infty((a, +\infty), (p, +\infty)) &= |a - p|, \\ d_\infty((-\infty, +\infty), (-\infty, +\infty)) &= \infty. \end{aligned}$$

Pontos em regiões diferentes, como (a, b) e $(p, +\infty)$, possuem distância igual a ∞ . Quando a distância for entre um ponto do plano à diagonal $\Delta = \{(x, x) \mid x \in \mathbb{R}\}$ ³

$$\begin{aligned} d_\infty((x, y), \Delta) &= \inf d_\infty((x, y), (a, a)) = \\ &= \max \left\{ \frac{(y - x)}{2}, \frac{(y - x)}{2} \right\} = \frac{1}{2}(y - x) \end{aligned}$$

Definição 3.22 (Distância Bottleneck). Sejam X e Y dois diagramas de persistência de mesma cardinalidade. Considere as bijeções $\eta: X \rightarrow Y$, tome o supremo das distâncias entre os pontos e suas imagens correspondentes segundo a norma L^∞ e o mínimo sobre todas as bijeções. Obtemos a **distância bottleneck** (distância de gargalo) entre os diagramas, por:

$$W_\infty(X, Y) = \inf_{\eta: X \rightarrow Y} \left\{ \sup_{x \in X} [d_\infty(x, \eta(x))] \right\}$$

A ideia da distância que acabamos de ver é que dois diagramas estão próximos um do outro se existe uma bijeção entre eles que não leva a um ponto muito longe.

Mostremos que a função, como definida, é de fato, uma métrica.

³A distância de um ponto ao conjunto é o ínfimo de todas as distâncias do ponto até os elementos do conjunto. No caso, a distância entre um ponto do plano à diagonal, será a distância do ponto até sua projeção ortogonal na diagonal.

- $W_\infty(X, X) = 0$ e $W_\infty(X, Y) > 0$

Se $W_\infty(X, Y) = 0$, significa que $\inf_{\eta: X \rightarrow Y} \left\{ \sup_{x \in X} [d_\infty(x, \eta(x))] \right\} = 0$ e ainda, que $\sup_{x \in X} [d_\infty(x, \eta(x))] = 0$.

Seja $x = (a, b) \in X$ e considere a bijeção $x \mapsto \eta(x) = (\eta a, \eta b)$, onde $(\eta a, \eta b)$ será a notação usada para as coordenadas do ponto da imagem de x por η .

Temos:

$$\sup_{x \in X} [d_\infty(x, \eta(x))] = \sup_{x \in X} [\max \{|a - \eta a|, |b - \eta b|\}] = 0$$

e ainda $|x - \eta(x)| \geq 0$, para cada $x \in X$.

Uma vez que tomamos o máximo das distâncias,

$$|a - \eta a| = 0 = |b - \eta b|,$$

portanto, $a = \eta a$ e $b = \eta b$.

Assim, como tomamos o máximo das diferenças e temos como resultado 0, segue que para todos os $x \in X$, $|x - \eta(x)| = 0$, e portanto, $x = \eta x$. Por tomarmos o ínfimo das bijeções, sucede que a bijeção η é a função identidade, $\eta = \text{Id}$.

Logo, $Y = \eta(X) = \text{Id}(X) = X$.

Reciprocamente, se $X = Y$, ao tomarmos o ínfimo das bijeções $\eta: X \rightarrow X$, teremos que $|x - \text{Id}(x)| = 0$ e o resultado segue.

- $W_\infty(X, Y) = W_\infty(Y, X)$

Como $\eta: X \rightarrow Y$ são bijeções, implica que existe $\eta^{-1}: Y \rightarrow X$ e, assim, $x = \eta^{-1}(y)$, para $x \in X$ e para algum $y \in Y$, daí

$$d_\infty(x, \eta(x)) = d_\infty(\eta^{-1}(y), \eta(\eta^{-1}(y))) = d_\infty(\eta^{-1}(y), y) = d_\infty(y, \eta^{-1}(y)),$$

pois L^∞ é métrica.

Portanto, $d_\infty(x, y) = d_\infty(y, x)$ e assim, temos o resultado.

- $W_\infty(X, Z) \leq W_\infty(X, Y) + W_\infty(Y, Z)$

Para a desigualdade triangular, sejam $\alpha: X \rightarrow Y$ e $\beta: Y \rightarrow Z$ bijeções.

Tome $\eta = \beta \circ \alpha: X \rightarrow Z$.

Assim,

$$\begin{aligned} d_\infty(x, \eta(x)) &= d_\infty(x, \beta(\alpha(x))) = d_\infty(x + \alpha(x), \beta(\alpha(x)) + \alpha(x)) \leq \\ &\leq d_\infty(x, \alpha(x)) + d_\infty(\alpha(x), \beta(\alpha(x))) \leq \\ &\leq \sup_{x \in X} d_\infty(x, \alpha(x)) + \sup_{y \in Y} d_\infty(y, \beta(y)), \end{aligned}$$

para todo $x \in X$.

Logo, como é para qualquer elemento de X ,

$$\sup_{x \in X} d_\infty(x, \beta(\alpha(x))) \leq \sup_{x \in X} d_\infty(x, \alpha(x)) + \sup_{y \in Y} d_\infty(y, \beta(y))$$

Agora, como a distância Bottleneck é o ínfimo por definição, temos que:

$$W_\infty(X, Z) \leq \sup_{x \in X} d_\infty(x, \beta(\alpha(x))) \leq \sup_{x \in X} d_\infty(x, \alpha(x)) + \sup_{y \in Y} d_\infty(y, \beta(y)),$$

para quaisquer bijeções $\alpha: X \rightarrow Y$ e $\beta: Y \rightarrow Z$.

Em particular,

$$W_\infty(X, Z) - \sup_{x \in X} d_\infty(x, \alpha(x)) \leq \sup_{y \in Y} d_\infty(y, \beta(y)),$$

para todo $\beta: Y \rightarrow Z$.

Tomando o ínfimo das bijeções $\beta: Y \rightarrow Z$ em ambos os lados,

$$W_\infty(X, Z) - \sup_{x \in X} d_\infty(x, \alpha(x)) \leq W_\infty(Y, Z),$$

para todo $\alpha: X \rightarrow Y$.

Por fim, consideramos

$$W_\infty(X, Z) - W_\infty(Y, Z) \leq \sup_{x \in X} d_\infty(x, \alpha(x)),$$

para todo $\alpha: X \rightarrow Y$.

Agora, tomando o ínfimo das bijeções $\alpha: X \rightarrow Y$,

$$W_\infty(X, Z) - W_\infty(Y, Z) \leq W_\infty(X, Y)$$

E portanto,

$$W_\infty(X, Z) \leq W_\infty(X, Y) + W_\infty(Y, Z),$$

como queríamos.

Assim, por W_∞ satisfazer todos os três axiomas de métrica, pode ser chamado de distância.

3.4 Imagem de Persistência

A distância Bottleneck é, computacionalmente, muito dispendiosa e também limitada, pois muitas técnicas de aprendizado de máquina, além das técnicas de agrupamentos, requerem mais do que uma matriz de distância de pares, como por exemplo a classificação de árvore de decisão e redes neurais. Além disso, uma representação vetorial dos diagramas de persistência possibilita o uso de métricas padrões como por exemplo, a métrica da soma (conhecida como métrica do táxi), a métrica euclidiana e a métrica infinito (3.21) (que é uma generalização da métrica do máximo). Essas medidas são mais eficientes, como poderemos ver no próximo capítulo.

Para então, fazer uma representação vetorial do diagrama de persistência, uma abordagem utilizada é tomar cada ponto do diagrama de persistência $(b, d) \in PD$ como o centro de uma distribuição normal bidimensional ⁴, centralizando uma Gaussiana em cada ponto e sobrepondo uma grade. Uma imagem é definida atribuindo a cada quadrado da grade

⁴Mais informações disponíveis em [13], página 45.

um valor de pixel. Formalmente, definimos $I(p)$, o valor de cada pixel p dentro de uma imagem, pela seguinte equação:

$$I(p) = \iint_p \left(\sum_{(b,d) \in PD} \frac{1}{2\pi\sigma_x\sigma_y} e^{-\frac{1}{2} \left(\frac{(x-b)^2}{\sigma_x^2} + \frac{(y-d)^2}{\sigma_y^2} \right)} \right) dydx, \quad (3.1)$$

onde σ_x e σ_y são as variância da gaussiana na direção x e y , respectivamente, e não é assumida nenhuma correlação cruzada.

Como o diagrama de persistência é um multiconjunto de pontos, permitimos várias gaussianas com o mesmo centro na soma.

Esta definição de pixel pondera igualmente as gaussianas definidas em pontos próximos à diagonal (que podem ser considerados como ruído) e distantes da diagonal (que representam um sinal forte). Essa ponderação pode eliminar pontos importantes do diagrama, caso a quantidade de pontos próximos a diagonal seja grande. Para contornar este problema, usa-se uma função peso que pode ser apresentada de variadas formas. Para os cálculos deste trabalho, usamos um parâmetro n , onde a imagem é construída segundo a persistênciaⁿ, que proporciona maior relevância aos elementos fora da diagonal.⁵

Exemplo 3.23. Considere o complexo K_0 utilizado na sessão anterior 3.7. Com auxílio do Software Geogebra, conseguimos efetuar o cálculo da somatória da equação (3.1) com o objetivo de visualizar a ideia da criação das imagens de persistência.

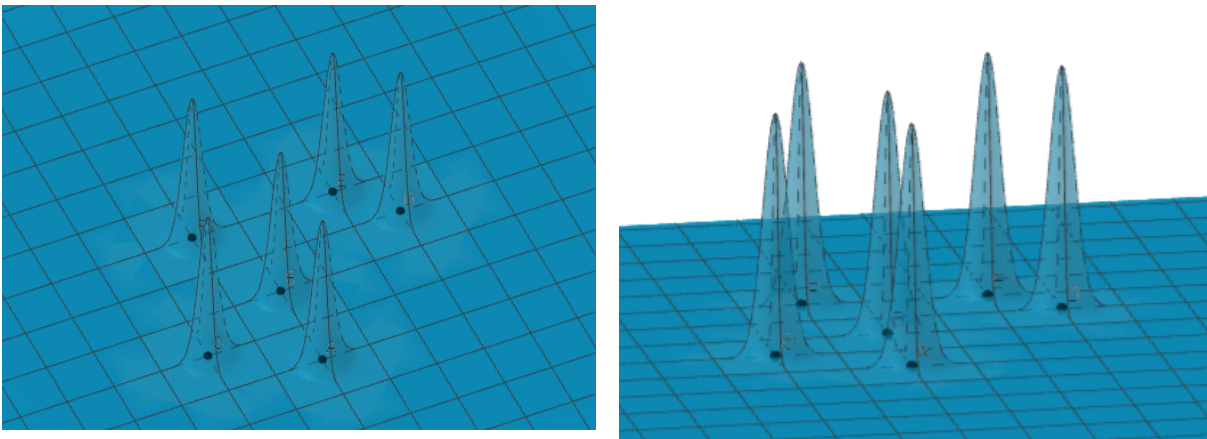


Figura 3.10: Diferentes perfis da somatória calculada para os pontos do gráfico.
Fonte: Própria autora.

⁵As informações desta seção foram retiradas do artigo [4].

4 Trabalho Desenvolvido

A clusterização de dados (agrupamento de dados), ganha holofotes com a informatização da vida, onde informações aos montes são disponibilizadas a cada segundo pelas redes. Além da problemática de dados falsos, a quantidade monstruosa dificulta uma análise assertiva sem a ajuda de ferramentas eletrônicas, que auxiliam na organização e filtragem desses dados.

Quando estes dados estão planilhados, há uma gama de possibilidades acessíveis para filtração e análise. Exemplos são o Microsoft Excel e, mais atualmente, o Power BI (*Bussiness Inteligence*). Porém, quando as informações têm formas mais elaboradas, como imagens, a análise também se mostra mais complexa.

Visto isto, o trabalho foi desenvolvido buscando possibilidades de análise e agrupamento para imagens em três dimensões (3D), utilizando ferramentas topológicas por meio da linguagem de programação **Python**.

As páginas seguintes conterão a descrição do *pipeline* (passo a passo) das atividades realizadas para interpretar as dados, as dificuldades do processo e resultados obtidos. Todas as imagens presentes no texto foram, ou geradas pelo programa desenvolvido, ou capturas de tela de arquivos produzidos.

4.1 Banco de dados

As imagens para a pesquisa são de domínio público e foram retiradas do banco de dados do curso Cos 429 de outono sobre Visão Computacional do Departamento de Informática, da Universidade de Princeton.[7]

São 200 arquivos com extensão “.ply” (*Polygon File Format* ou *Stanford Triangle Format*)¹, de variados tamanhos, divididos em 20 categorias: *biplane* (avião biplano), *desk chair* (cadeira de escritório), *dining chair* (cadeira de jantar), *fighter jet* (avião a jato), *fish* (peixe), *flying bird* (pássaro), *guitar* (violão), *handgun* (revólver), *head* (cabeça), *helicopter* (helicóptero), *human armns out* (braços humanos), *human* (humanos), *potted plant* (plantas em vaso), *race car* (carro de corrida), *sedan* (carro sedan), *shelves* (prateleiras), *ship* (navio), *sword* (espada), *table* (mesa) e *vase* (vaso).

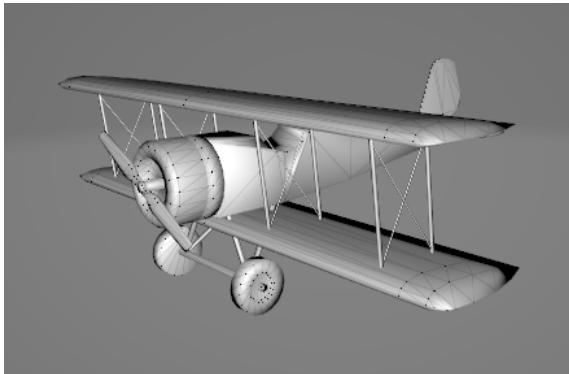
Acrescentei à esta miscelânea, o desenho de um cubo, feito por mim no programa Blender², para auxiliar nos testes, entendimento do processo e elaboração do programa,

¹“O padrão PLY foi desenvolvido na Universidade de Stanford [...] como um meio de armazenar dados 3D simples [...]. O arquivo PLY armazena objetos como uma lista de polígonos planos que podem ter as seguintes propriedades definidas: cor e transparência, descrição da superfície e coordenadas espaciais.” [12]

²“[...] é um programa de computador de código aberto, desenvolvido pela Blender Foundation, para modelagem, animação, texturização, composição, renderização, edição de vídeo e criação de aplicações interativas em 3D.” [17]

já que por ser simples, possibilita verificações manuais.

Na Figura 4.1 podemos ver alguns exemplos de imagens do banco de dados utilizado.



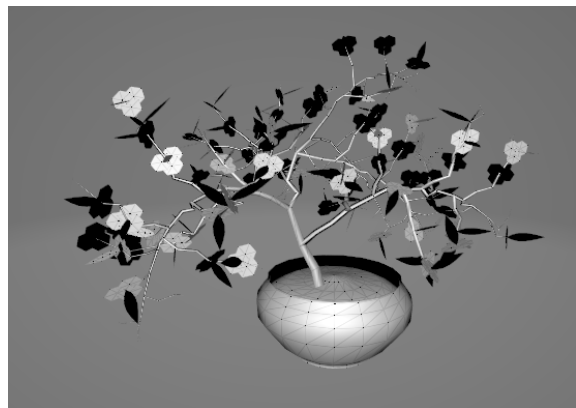
(a) Biplane 0



(b) Guitar 9



(c) Untitled 3



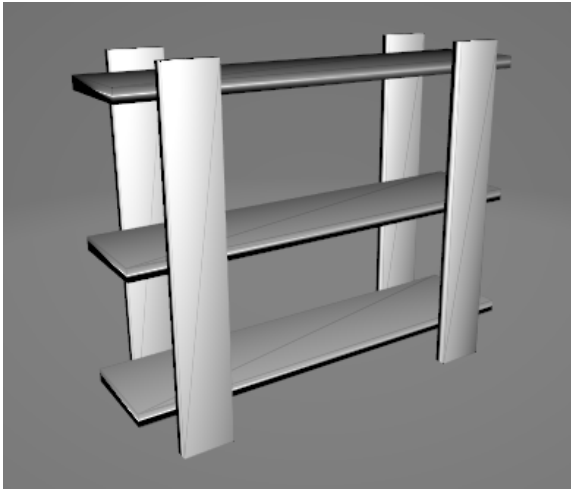
(d) Potted plant 1

Figura 4.1: Exemplos de imagens do banco de dados.

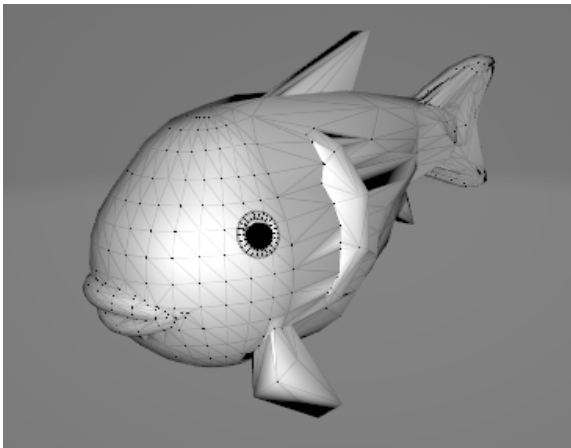
Fonte: Captura de tela pelo Fotos.

É possível perceber, observando as imagens, pequenos pontos na estrutura. São eles, que sustentam os objetos e são a fonte de informação utilizada para as análises.

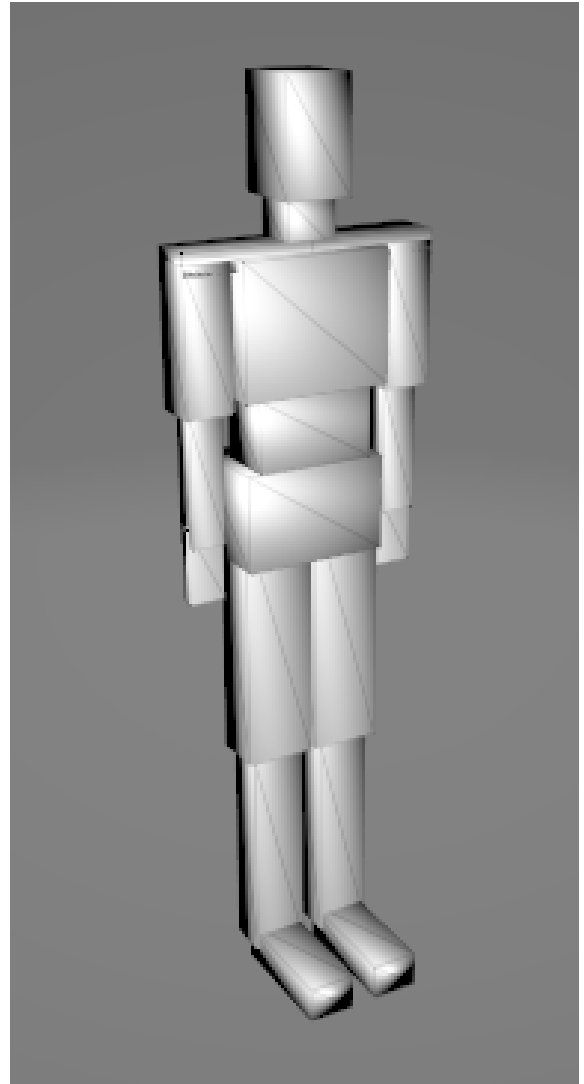
Também se nota que, para que as formas tenham aparência mais arredondada, elas precisam de uma quantidade maior de pontos, visto que as imagens são formadas por segmentos de retas ligando os pontos, e não curvas (Figura 4.2).



(a) Shelves 7, 56 pontos.



(b) Fish 7, 4.117 pontos.



(c) Human 8, 148 pontos.

Figura 4.2: Imagens com quantidade de pontos diferentes.

Fonte: Captura de tela pelo Fotos.

4.2 Diagrama de Persistência

Após o reconhecimento das imagens e categorização por meio de dicionários, as imagens originalmente em formato `.ply` foram convertidas para o formato matricial, onde as colunas equivalem aos eixos x , y e z e as linhas variam de acordo com a quantidade de pontos de cada figura.

As informações obtidas foram armazenadas em um dicionário, no formato `{Imagem: (matriz de pontos, quantidade de pontos)}`.

```
1 pasta = '...' # local onde as imagens estão armazenadas
2
3 arquivos = '...' # lista de nomes dos arquivos
4
```

```

5  pcds = {} # dicionário para armazenar a matriz de pontos da imagem e a
   ↪ quantidade de pontos
6
7  for i in arquivos:
8      pcd = o3d.io.read_point_cloud(pasta+i) # lê a nuvem de pontos dos
   ↪ arquivos
9      pontos = np.asarray(pcd.points) # converte o formato open3d para
   ↪ numpy array, (n,3)=(linhas, colunas: eixos x, y e z)
10     ponto = np.unique(pontos, axis=0) # pontos sem repetição, axis=0:
   ↪ percorre linhas, axis=1: percorre colunas
11     quantidade = len(ponto) # quantidade de pontos do arquivo, sem
   ↪ repetição
12
13     pcds.update({i: (ponto, quantidade)})

```

Programa 4.1: Programa para gerar as matrizes de pontos e armazenar as informações.

Exemplo 4.1. Matriz de pontos e quantidade de pontos do Cubo 4.1c

```

1  print(pcds['untitled3.ply'])

```

```

(array([[ -1.52003551,  0.57333529,  0.60064852],
        [ -1.180071, -1.26249039, -0.11640674],
        [ -0.51576865,  1.36078799, -0.93927568],
        [ -0.17580414, -0.47503772, -1.65633094],
        [  0.17580414,  0.47503772,  1.65633094],
        [  0.51576865, -1.36078799,  0.93927568],
        [  1.180071,  1.26249039,  0.11640674],
        [  1.52003551, -0.57333529, -0.60064852]]), 8)

```

Para a análise topológica dos dados, por meio da homologia, os espaços são dados por complexos, que possibilitam extrair informações geométricas.

Neste ponto, a informação sobre as imagens são vetores com entradas tridimensionais (valores para x , para y e z , do plano cartesiano) – as coordenadas dos pontos. Para a criação dos complexos, as informações atuais foram filtradas pelo método de Vietoris-Rips (3.1.2) e armazenados. Estes complexos criados pela filtração são utilizados no cálculo do diagrama de persistência por meio da ferramenta **Rips** do pacote **ripser** disponível para Python.

O diagrama foi calculado em H_1 com coeficientes no corpo $\mathbb{Z}_2 = \{0, 1\}$. A escolha do H_1 foi visando melhores resultados, visto que H_0 , que são o número de componentes conexas, retornaria a quantidade de pontos e a escolha do corpo, para simplificar os cálculos - o que exige menos para o processamento dos dados.

```

1  def H1(arquivo):
2
3      rips = Rips(maxdim=1, coeff=2) # maxdim: H_1, coeff: Z_2

```

```

4     diagrama_h1 = rips.fit_transform(arquivo)[1] # [1]:  $H_1$ , [0]:  $H_0$  e
      ↪ sem, calcula  $H_0$  e  $H_1$ 
5
6     return diagrama_h1

```

Programa 4.2: Função que calcula H_1 de um arquivo, por meio de sua matriz de pontos.

O cálculo do diagrama de persistência resulta em pares ordenados com informações de (nascimento, morte) dos simplexes de acordo com a filtração do complexo, que neste caso, é dado pela nuvem de pontos da imagem. 3.15

Devido a morosidade do cálculo do diagrama de persistência para imagens com muitos pontos e a limitação da máquina³, os diagramas foram gerados em blocos, de acordo com a quantidade de pontos das imagens: de 0 à 999 pontos; 1.000 à 1.999 pontos; 2.000 à 2.999 pontos e de 3.000 à 4.999.

```

1  # Gerar o diagrama de persistência com menos de 1000 pontos
2
3  pasta_pd = 'caminho da pasta onde os arquivos PDs serão salvos'
4
5  arquivos_menores = []
6
7  for j in pcids.keys():
8      if pcids[j][1] < 1000:
9          arquivos_menores.append(j)
10         pd = H1(pcids[j][0])
11
12         # salvar o diagrama em um csv
13         pd_file = 'pd_'+j.split('.')[0]+' .csv' # string do nome do
          ↪ arquivo a ser criado
14         np.savetxt(pasta_pd+pd_file, pd, delimiter=',')

```

Programa 4.3: Programa para gerar os PDs das imagens com até 1.000 pontos.

Após a iteração para todas as imagens que seguiam o parâmetro posto, obteve-se uma amostra de 158 imagens, onde 100 possuem até 999 pontos, 31 de 1.000 – 1.999, 6 de 2.000 – 2.999 e, as demais 21 pertencem ao bloco de 3.000 – 4.999 pontos.

Em razão desta “filtragem”, os nomes das imagens utilizadas para o estudo foram armazenados para utilização futura, como é possível observar na linha 5 do Programa 4.3.

³Os programas foram executados pelo “Google Colaboratory”. Também conhecido como “Colab”, é um produto do Google Research, área de pesquisas científicas do Google. **O Colab permite que qualquer pessoa escreva e execute código Python arbitrário pelo navegador** e é especialmente adequado para aprendizado de máquina, análise de dados e educação. Mais tecnicamente, o Colab é um serviço de notebooks hospedados do Jupyter que não requer nenhuma configuração para usar e oferece acesso sem custo financeiro a recursos de computação como GPUs.” [5](grifo nosso)

Exemplo 4.2. Pares de pontos do diagrama de persistência do Cubo 4.1c.

```
1 H1(pcds['untitled3.ply'][0])
```

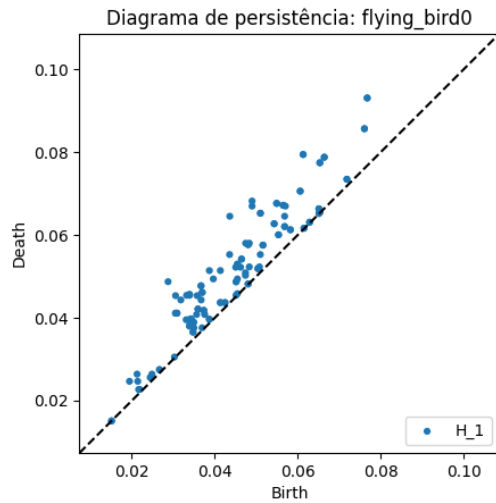
```
Rips(maxdim=1, thresh=inf, coeff=2, do_cocycles=False, n_perm = None,
↪ verbose=True)
array([[2.0, 2.82842708],
       [2.0, 2.82842708],
       [2.0, 2.82842708],
       [2.0, 2.82842708],
       [2.0, 2.82842708]])
```

Observando os exemplos 4.1 e 4.2, nota-se que os 8 pontos do cubo formam uma cadeia de 5 complexos, que nascem e morrem no mesmo nível de filtração. Na imagem 4.4, as barras de H_1 , que são 5, estão sobrepostas - por apresentarem o mesmo valor - por isso a aparência é de existir apenas um ponto.

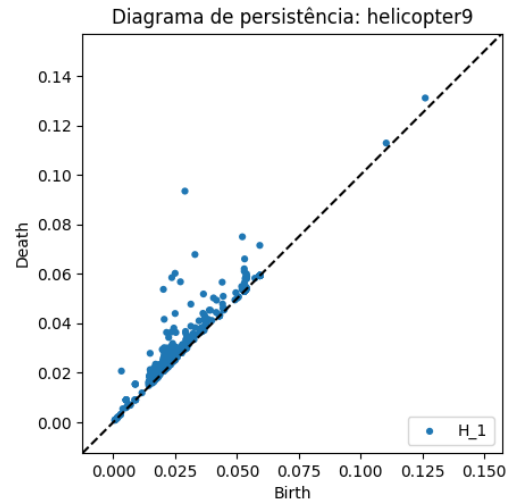
Como vimos em 3.8b, os pares de pontos da persistência de um complexo filtrado podem ser plotados em um Diagrama, chamado **Diagrama de Persistência**. No Programa 4.4 podemos observar como o diagrama pode ser executado em Python, utilizando a ferramenta plot.

```
1 def PD(arquivo, pontos_pd, salvar=False, caminho_salvar=''):
2
3     if salvar and caminho_salvar == '':
4         print('Para salvar é necessário escolher um caminho')
5         return None
6
7     rips = Rips(maxdim=1, coeff=2) # calcula o diagrama de persistência
8     rips.plot(pontos_pd, labels='H_1') # plota o diagrama de persistência
9     plt.title('Diagrama de persistência:
↪ {}'.format(arquivo.split('.')[0]))
10    if salvar:
11        plt.savefig(caminho_salvar+arquivo.split('.')[0]+' .png')
12    gráfico = plt.show()
13
14    return gráfico
```

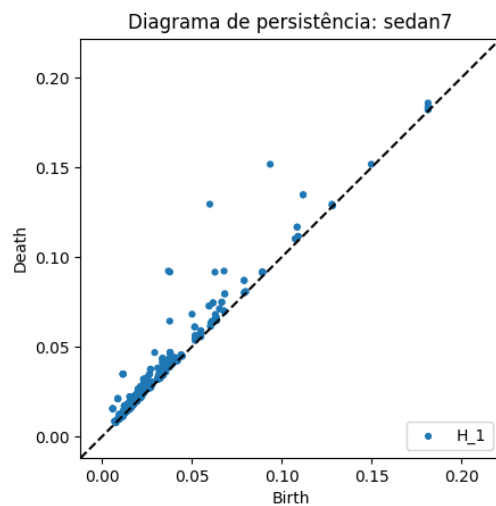
Programa 4.4: Função para exibir o diagrama de persistência de uma imagem.



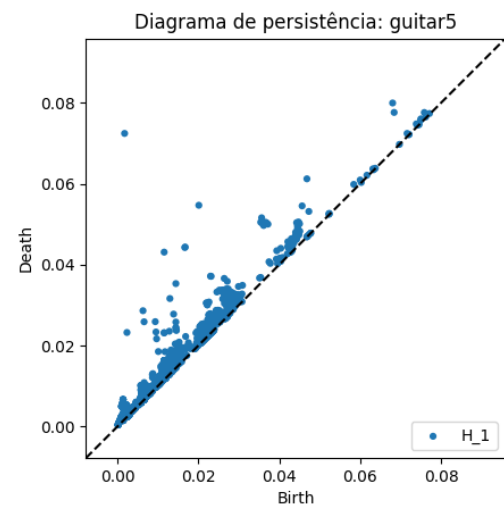
(a) Flying Bird 0, 233 pontos.



(b) Helicopter 9, 1.201 pontos.



(c) Sedan 7, 2.170 pontos.



(d) Guitar 5, 4.694 pontos.

Figura 4.3: PDs de imagens com diferente quantidade de pontos

Exemplo 4.3. Diagrama de persistência do Cubo 4.1c.

```

1 pontos_pd_cubo=np.array([[2, 2.82842708], [2, 2.82842708], [2,
  ↳ 2.82842708], [2, 2.82842708], [2, 2.82842708]])
2
3 PD('untitled3.ply', pontos_pd_cubo, salvar=True, caminho_salvar='onde
  ↳ salvar a imagem')
```

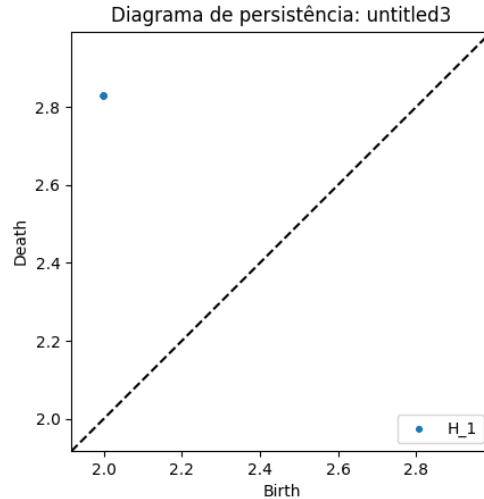


Figura 4.4: Diagrama de Persistência do Cubo.

4.3 Imagem de Persistência

Como visto na seção 3.4, do capítulo anterior, a imagem de persistência (PI) é uma forma de representar as informações dos pares de persistência de um complexo, por meio de um vetor.

Para gerar a imagem utilizamos o pacote `PersistenceImager` da biblioteca `persim`. Ele permite a escolha de parâmetros para a criação das imagens, são eles:

- **Resolução da Imagem:** a resolução corresponde ao tamanho do pixel, ao tamanho da grade quadriculada que será posta sobre a imagem.
- **Distribuição:** é o método de distribuição de probabilidade que será associado aos pontos do diagrama de persistência.
- **Ponderação:** a ponderação é usada para ajustar os dados de forma a obter representações mais fidedignas. No diagrama de persistência, os pontos próximos à diagonal são considerados ruídos, isto é, têm pequena relevância para os resultados; e, se em grande número, podem maquiar a imagem, destacando pontos não importantes. Assim, a ponderação permite que pontos com maior persistência ganhem pesos maiores para ganhar destaque nas imagens.

O programa irá analisar os dados de persistência segundo os parâmetros escolhidos e gerar uma matriz de pontos (o tamanho da matriz está relacionado ao tamanho do pixel, tamanho dos pontos e tamanho dos eixos escolhidos). Posteriormente, os dados da matriz são renderizados (processados digitalmente), resultando em uma imagem pseudo-colorida⁴.

⁴A imagem pseudo-colorida relaciona cores à valores, segundo um fatiamento de intensidade. [19]

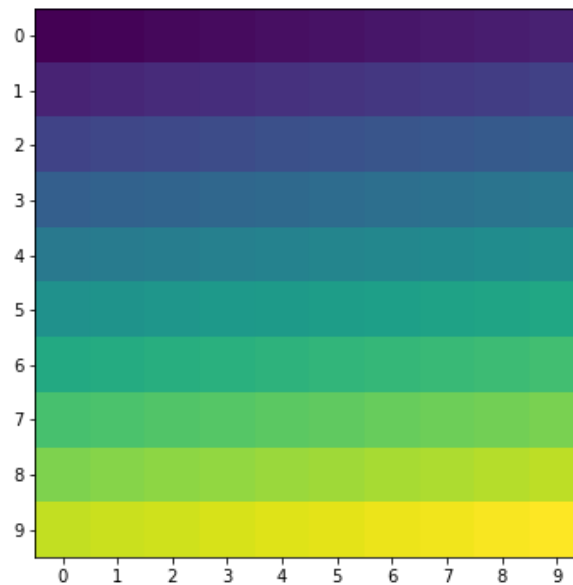


Figura 4.5: Escala de cor utilizada para colorir as imagens.

```

1 def PI_peso(caminho_pasta_arquivos, nome_arquivo, pixel, n,
2             salvar=False, caminho_salvar=''):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     arquivos = caminho_pasta_arquivos
9     string = nome_arquivo
10    sigma = 0.5
11    n = n
12    p = pixel
13
14    nome = arquivos+string
15    a = np.loadtxt(nome, delimiter=',')
16    a = a*100
17
18    maior = max(max(a[:, 0]), max(a[:, 1]))
19
20    pimgr = PersistenceImager(pixel_size=p, weight_params={'n': n},
21                             kernel_params={'sigma': [[sigma, 0],
22                                                    [0, sigma]]}, birth_range=(0, maior*1.10),
23                             pers_range=(0, maior*1.10))
24    ping = pimgr.transform(a)
25    if salvar:
26        np.savetxt(caminho_salvar+'nome_salvar'+'.csv', ping,
27                  delimiter=',')

```

```

28
29     # diagrama de persistência
30
31     rips = Rips(maxdim=1, coeff=2)
32
33     # persistência x nascimento
34
35     pimgr.plot_diagram(a)
36     if salvar:
37         plt.savefig(caminho_salvar+'nome_salvar'.png')
38     plt.show()
39
40     # gerar imagem de persistência
41
42     plt.title('Imagem de Persistência: {}'.format(string))
43     pimgr.plot_image(pimg)
44     if salvar:
45         plt.savefig(caminho_salvar+'nome_salvar'+'.png')
46     plt.show()

```

Programa 4.5: Função para gerar o diagrama de persistência (nascimento, persistência) e imagem de persistência com peso.

```

1  def PI(caminho_pasta_arquivos, nome_arquivo, pixel, salvar=False,
2         caminho_salvar=''):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     arquivos = caminho_pasta_arquivos
9     string = nome_arquivo
10    p = pixel
11
12    nome = arquivos+string
13    a = np.loadtxt(nome, delimiter=',')
14    a = a*100
15
16    maior = max(max(a[:, 0]), max(a[:, 1]))
17
18    pimgr = PersistenceImager(pixel_size=p, birth_range=(0, maior*1.10),
19                              pers_range=(0, maior*1.10))
20    pimg = pimgr.transform(a)
21    if salvar:
22        np.savetxt(caminho_salvar+'nome_salvar'.csv', pimg,
23                  delimiter=', ')
24
25    # gerar imagem de persistência
26

```

```

27 plt.title('Imagem de Persistência: {}'.format(string))
28 pimgr.plot_image(pimg)
29 if salvar:
30     plt.savefig(caminho_salvar+'nome_salvar'+'.png')
31 plt.show()

```

Programa 4.6: Função para gerar a imagem de persistência.

Nesta pesquisa, foi usada uma resolução de 0.1 pixel, com Distribuição Gaussiana. Foram geradas imagens com e sem ponderação para comparação dos resultados obtidos.

É importante observar que, diferentemente do Diagrama de Persistência, as imagens são construídas pelo par (nascimento, persistência). Por esta razão, foram construídos diagramas com este par para facilitar a comparação dos resultados obtidos. As imagens geradas pelo programa podem ser conferidas em 4.6, 4.7, 4.8 e 4.9.

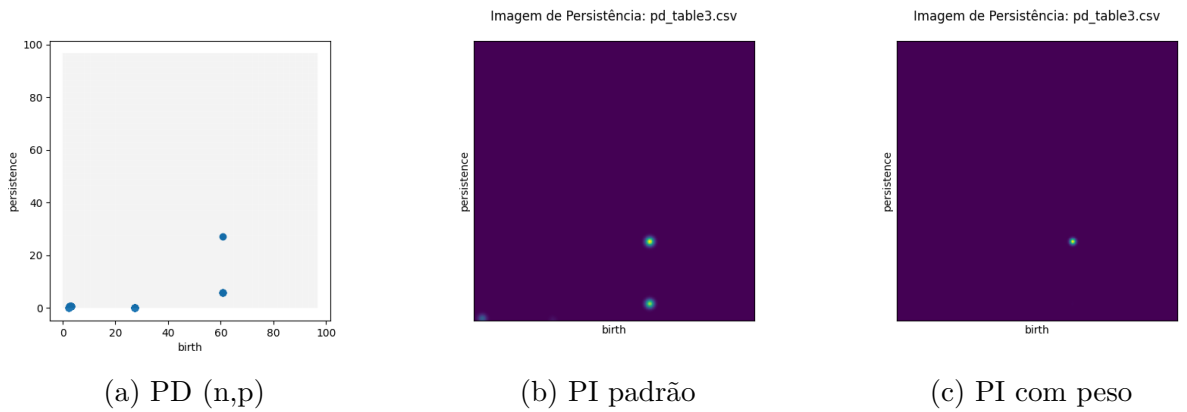


Figura 4.6: Table 3, 56 pontos.

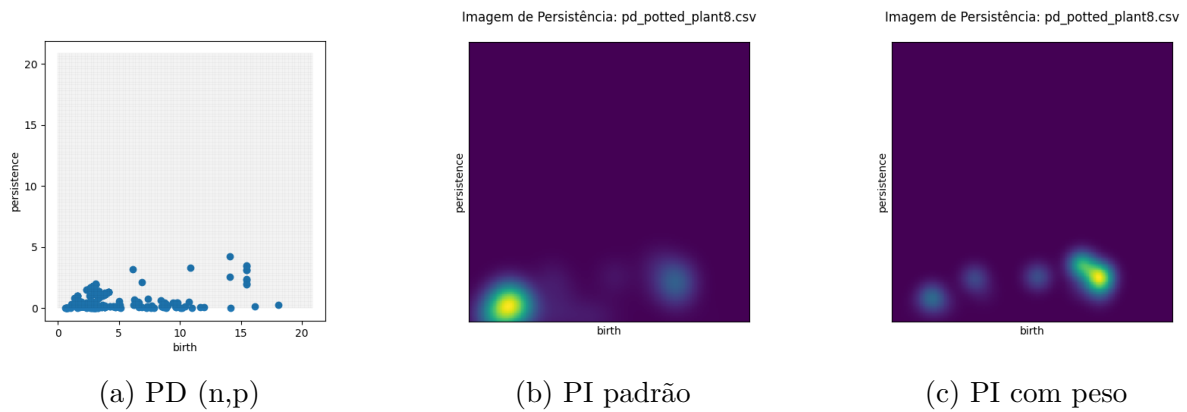


Figura 4.7: Potted Plant 8, 1.325 pontos.

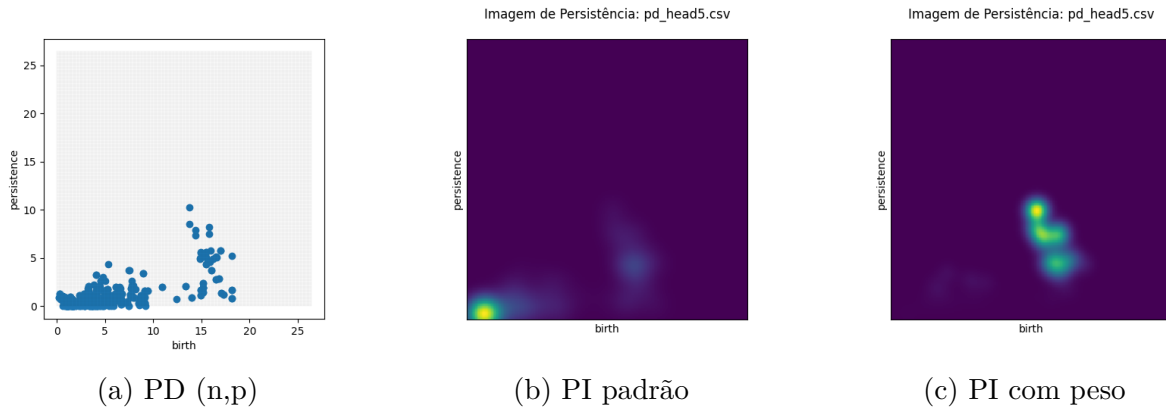


Figura 4.8: Head 5, 2.034 pontos.

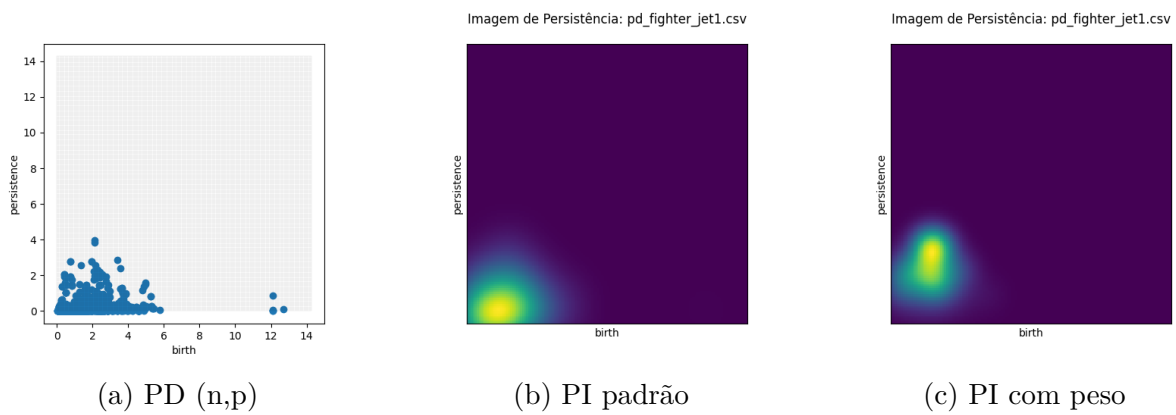
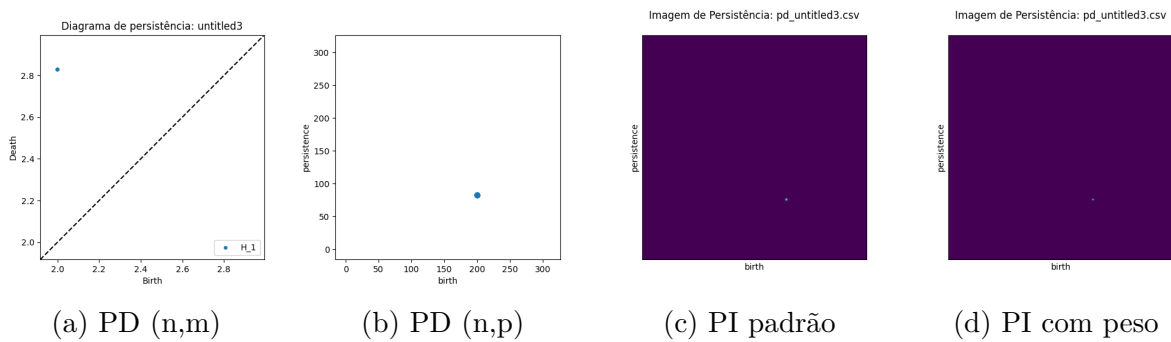


Figura 4.9: Fighter Jet 1, 4.427 pontos.

Exemplo 4.4. Imagem de persistência do Cubo (4.1c).



4.4 Distância entre os Diagramas de Persistência

4.4.1 Distância Bottleneck

Como visto na Seção 3.3, essa distância encontra o mínimo de todas as bijeções possíveis entre todos os pontos. Ela foi usada para comparar a distância entre os diagramas de persistência. Como as imagens possuem um número grande de pontos, seus respectivos

diagramas também contém. Logo, calcular todas alternativas, foi um processo extremamente demorado, levando 15.818,04 minutos, isto é, cerca de 10,98 dias.

```
1 pasta_pd = '...' # caminho da pasta dos arquivos PDs
2
3 pasta = os.listdir(pasta_pd)
4 t = len(pasta)
5
6 distâncias = np.zeros((t, t))
7
8 tempos = np.zeros((t, t))
9
10 for i in [10]: # a linhas foram iteradas de forma manual
11     for j in range(i+1, t):
12
13         print(i, j)
14
15         s_time = time.process_time()
16
17         a = os.listdir(pasta_pd)[i]
18         b = os.listdir(pasta_pd)[j]
19
20         print('Distância de %s até %s' % (a, b))
21
22         a1 = np.loadtxt(pasta_pd+a, delimiter=',')
23         b1 = np.loadtxt(pasta_pd+b, delimiter=',')
24
25         distance_bottleneck, matching = persim.bottleneck(a1, b1,
26                                                         matching=True)
27
28         distâncias[i, j] = distance_bottleneck
29
30         e_time = time.process_time()
31         tempos[i, j] = e_time-s_time
32
33         print('*Distância: %.4f, tempo de execução: %.2f segundos.' %
34               (distâncias[i, j], tempos[i, j]))
35
36     # aqui salvar as distâncias em um csv
37
38     f = open('caminho do arquivo', 'a') # (nome do arquivo a ser
39     ↪ criado/aberto, r=read / a=append)
40     np.savetxt(f, distâncias[i], newline=',') # (onde salvar, o que
41     ↪ acrescentar, como separar a próxima informação - se não colocar
42     ↪ nada vai pra próxima linha)
43     f.write("\n") # encerra a linha que foi acrescentada e cria uma nova
44     f.close() # fecha o arquivo
45
46     # aqui salvar o tempo de execução em um csv
```

```

45 g = open('caminho do arquivo', 'a')
46 np.savetxt(g, tempos[i], fmt='%.2f', newline=',')
47 g.write("\n")
48 g.close()

```

Programa 4.7: Calcular a distância bottleneck entre os Diagramas de Persistência.

A ideia foi comparar a distância entre todos os diagramas de persistência obtidos, ou seja, cada diagrama foi comparado com outros 157. Para diminuir o tempo de processamento, as informações foram armazenadas em uma Matriz_{158x158} superior, iteradas manualmente linha à linha utilizando o Programa 4.7 e exportadas para um arquivo de texto (.csv)⁵.

Após feito, organizou-se as informações tornando a matriz simétrica e referenciando o nome das imagens de acordo com as distâncias calculadas, por meio das funções 4.8 e 4.9 definidas abaixo, resultando na matriz da Figura 4.11.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_ch
2	biplane0.ply	0	0.03646799549	0.04164369404	0.02724721283	0.02724721283	0.02545116842	0.07001279294	0.02724721283	0.03985600173	0.02
3	biplane3.ply	0.03646799549	0	0.03646799549	0.03646799549	0.03646799549	0.03646799549	0.07001279294	0.03646799549	0.04432789981	0.03
4	biplane4.ply	0.04164369404	0.03646799549	0	0.03620638698	0.02662814781	0.0402328074	0.07001279294	0.04498723149	0.04432789981	0.04
5	biplane6.ply	0.02724721283	0.03646799549	0.03620638698	0	0.01583196223	0.02470792271	0.07001279294	0.01816821657	0.04432789981	0.01
6	biplane7.ply	0.02724721283	0.03646799549	0.02662814781	0.01583196223	0	0.02091150917	0.06807938218	0.02572518401	0.04432789981	0.02
7	biplane8.ply	0.02545116842	0.03646799549	0.0402328074	0.02470792271	0.02091150917	0	0.07001279294	0.02411514893	0.0376214534	0.02
8	biplane9.ply	0.07001279294	0.07001279294	0.07001279294	0.07001279294	0.06807938218	0.07001279294	0	0.07001279294	0.07001279294	0.07
9	desk_chair1.ply	0.02724721283	0.03646799549	0.04498723149	0.01816821657	0.02572518401	0.02411514893	0.07001279294	0	0.04432789981	0.01
10	desk_chair2.ply	0.03985600173	0.04432789981	0.04432789981	0.04432789981	0.04432789981	0.0376214534	0.07001279294	0.04432789981	0	0.04
11	desk_chair5.ply	0.02288037539	0.03646799549	0.04539142549	0.01909016073	0.02682198584	0.02470792271	0.07001279294	0.01909016073	0.04432789981	
12	desk_chair6.ply	0.05086344481	0.04995788634	0.05086344481	0.05086344481	0.05086344481	0.04549570382	0.07001279294	0.05086344481	0.03158343102	0.04
13	desk_chair7.ply	0.06069895625	0.06069895625	0.06069895625	0.06069895625	0.06069895625	0.06069895625	0.06912180781	0.06069895625	0.06069895625	0.04
14	desk_chair8.ply	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1136096343	0.1
15	dining_chair0.ply	0.06050176919	0.06050176919	0.06050176919	0.06050176919	0.06050176919	0.06050176919	0.02282866836	0.06050176919	0.06050176919	0.06
16	dining_chair1.ply	0.03331854939	0.03646799549	0.04576878622	0.03331854939	0.03331854939	0.03331854939	0.07001279294	0.05804480612	0.05804480612	0.05
17	dining_chair2.ply	0.05804480612	0.05804480612	0.05804480612	0.05804480612	0.05804480612	0.05804480612	0.06349188089	0.03331854939	0.04432789981	0.03
18	dining_chair3.ply	0.02724564075	0.03646799549	0.04009845108	0.02724564075	0.02732454799	0.02724564075	0.07001279294	0.04588162899	0.04588162899	0.04
19	dining_chair4.ply	0.02724721283	0.03646799549	0.04576878622	0.02095897496	0.02919058036	0.02095897496	0.07001279294	0.02095897496	0.04432789981	0.02
20	dining_chair5.ply	0.04588162899	0.04588162899	0.04588162899	0.04588162899	0.04588162899	0.04588162899	0.06944400072	0.02724564075	0.04432789981	0.02
21	dining_chair6.ply	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08853635192	0.08

Figura 4.11: Recorte da matriz de distância Bottleneck.

```

1 def simetrica(caminho_arquivo, salvar=False):
2
3     arquivo = np.genfromtxt(caminho_arquivo, delimiter=',')[:, :-1] #
4     ↪ [linhas, columnas]
5
6     matriz_simétrica = arquivo+arquivo.T
7
8     # salvar o arquivo alterado
9     if salvar:
10         np.savetxt(caminho_arquivo, matriz_simétrica, delimiter=',')

```

⁵“Um arquivo CSV (Valores Separados por Vírgula) é um tipo especial de arquivo que [...] em vez de armazenar informações em colunas, [...] armazenam informações separadas por vírgulas” [8]

```
10
11     return matriz_simétrica
```

Programa 4.8: Função para tornar uma matriz simétrica quando precisa excluir última coluna vazia.

```
1  def rotulos(caminho_arquivo, caminho_nome, salvar=False):
2
3      nome = np.loadtxt(caminho_nome, delimiter=',', dtype=str)
4      arquivo = pd.read_csv(caminho_arquivo, sep=',', header=None)
5
6      arquivo.columns = nome # inserir nome das colunas
7      arquivo.index = nome # inserir coluna de índices
8
9      if salvar:
10         arquivo.to_csv(caminho_arquivo, index=True)
11
12     return arquivo
```

Programa 4.9: Função para inserir nome das colunas e índices, em uma matriz.

Por meio da matriz, é possível conhecer a distância topológica entre quaisquer imagens do estudo. Porém, como a matriz é extensa, não está claro quais delas são mais ou menos próximas.

Para facilitar a análise, criou-se uma função que filtra as imagens que são mais próximas, como definido no Programa 4.10. Na distância bottleneck, o parâmetro de distância máxima escolhido foi de 0,01. Teste com números maiores, trouxeram uma lista de tamanho considerável e com medida 0,001, retornou uma lista vazia, ou seja, nenhuma imagem tem distância menor do que 0,001. Parte da matriz resultante pode ser observada na imagem 4.12.

```
1  def filtro(caminho_arquivo, parâmetro):
2
3      arquivo = pd.read_csv(caminho_arquivo, delimiter=',', header=None)
4
5      nome_novo = caminho_arquivo[:-4]
6      nome_parâmetro = str(parâmetro)
7      arquivo_novo = nome_novo+' menor que '+nome_parâmetro+'.csv'
8
9      n = np.size(arquivo, 0)
10
11     # criar arquivo novo com uma linha
12
13     vetor = np.arange(1, n+1)
14
15     g = open(arquivo_novo, 'a+')
16     np.savetxt(g, vetor, newline=',', fmt='%d')
17     g.write('\n')
18     g.close()
```

```
19
20     # filtrar distâncias menores que um parâmetro
21
22     parecidos = {}
23
24     for i in range(1, n):
25         nome_linha = arquivo.iloc[i, 0]
26         for j in range(1, n-1):
27             distância = arquivo.iloc[i, j]
28             nome_coluna = arquivo.iloc[0, j]
29             if float(distância) < parâmetro and float(distância) != 0:
30                 if nome_linha in parecidos.keys():
31                     parecidos[nome_linha].append(nome_coluna)
32                 else:
33                     parecidos[nome_linha] = [nome_coluna]
34             else:
35                 continue
36
37     # salvar o nome dos arquivos em um csv
38
39     if nome_linha in parecidos.keys():
40         g = open(arquivo_novo, 'a+')
41         np.savetxt(g, parecidos[nome_linha], fmt='%s', newline=',')
42         g.write('\n')
43         g.close()
44
45     print(parecidos)
46
47     # excluir colunas vazias e acrescentar o índice no arquivo novo
48
49     arquivo2 = pd.read_csv(arquivo_novo, delimiter=',')
50
51     arquivo2 = arquivo2.dropna(how='all', axis=1)
52     arquivo2.index = parecidos.keys()
53
54     arquivo2.to_csv(arquivo_novo, index=True)
```

Programa 4.10: Função para filtrar as imagens mais próximas.

	A	B	C	D	E	F	G	H	I
1		1	2	3	4	5	6	7	8
2	biplane6.ply	fighter_jet9.ply	ship0.ply	human3.ply					
3	desk_chair1.ply	flying_bird4.ply	sword1.ply						
4	fighter_jet9.ply	biplane6.ply							
5	fish5.ply	flying_bird4.ply	guitar8.ply	helicopter8.ply	human_arms_out7.p	ship0.ply	ship2.ply	helicopter0.ply	ship8.ply
6	fish8.ply	ship2.ply	sword0.ply	sword7.ply	sword8.ply	helicopter0.ply			
7	flying_bird0.ply	flying_bird5.ply	ship0.ply	ship2.ply	helicopter3.ply	human3.ply			
8	flying_bird4.ply	desk_chair1.ply	fish5.ply	sword0.ply	sword5.ply	sword6.ply	sword7.ply	sword8.ply	handgun0.ply
9	flying_bird5.ply	flying_bird0.ply	helicopter3.ply						
10	guitar8.ply	fish5.ply	ship0.ply	ship2.ply	flying_bird7.ply				

Figura 4.12: Recorte da matriz de imagens que possuem distância bottleneck menor do que 0,01.

Objetivando comparar visualmente estas imagens ditas parecidas, imagens de seus pontos foram gerados por meio da função descrita no programa 4.11.

```

1 def grafico(caminho_arquivo, nome_arquivo, caminho_pasta_imagens,
2             salvar=False, caminho_salvar='', tipo=''):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     arquivo = pd.read_csv(caminho_arquivo)
9     n = len(arquivo.columns)
10
11     linha = arquivo.loc[arquivo.iloc[:, 0] == nome_arquivo] #
12     ↪ arquivo.iloc[linha,coluna]
13
14     for j in range(0, n):
15         nome = str(linha.iloc[0, j])
16         if nome != 'nan':
17             pcd = o3d.io.read_point_cloud(caminho_pasta_imagens+'/'+nome)
18             ↪ # ler a nuvem de pontos "point cloud" dos arquivos
19             pontos = np.asarray(pcd.points)
20             fig = px.scatter_3d(x=pontos[:, 0], y=pontos[:, 1],
21                                z=pontos[:, 2], title=nome)
22             fig.update_layout(scene_camera=dict(up=dict(x=0, y=2, z=0),
23             center=dict(x=0, y=0, z=0),
24             eye=dict(x=-2, y=1, z=2.5)),
25             scene=dict(xaxis=dict(
26             backgroundcolor='white',
27             color='black', gridcolor='#f0f0f0',
28             title_font=dict(size=8),
29             tickfont=dict(size=8)),
30             yaxis=dict(backgroundcolor='white',
31             color='black', gridcolor='#f0f0f0',
32             title_font=dict(size=8),
33             tickfont=dict(size=8)),
34             zaxis=dict(backgroundcolor='white',
35             color='black', gridcolor='#f0f0f0',

```

```

34         title_font=dict(size=8),
35         tickfont=dict(size=8)))
36     fig.update_traces(marker=dict(size=3,
37                               line=dict(color='black',width=0.1)))
38     fig.update(layout_coloraxis_showscale=False)
39     if salvar and tipo == 'png':
40         fig.write_image('nome_imagem'+'.png')
41     if salvar and tipo == 'html':
42         fig.write_html('nome_imagem'+'.html')
43     fig.show()

```

Programa 4.11: Função que exibe a nuvem de pontos das imagens filtradas.

Foram escolhidas as imagens próximas à Fighter jet 9, linha 4 da tabela 4.12 para serem geradas, como exemplo.

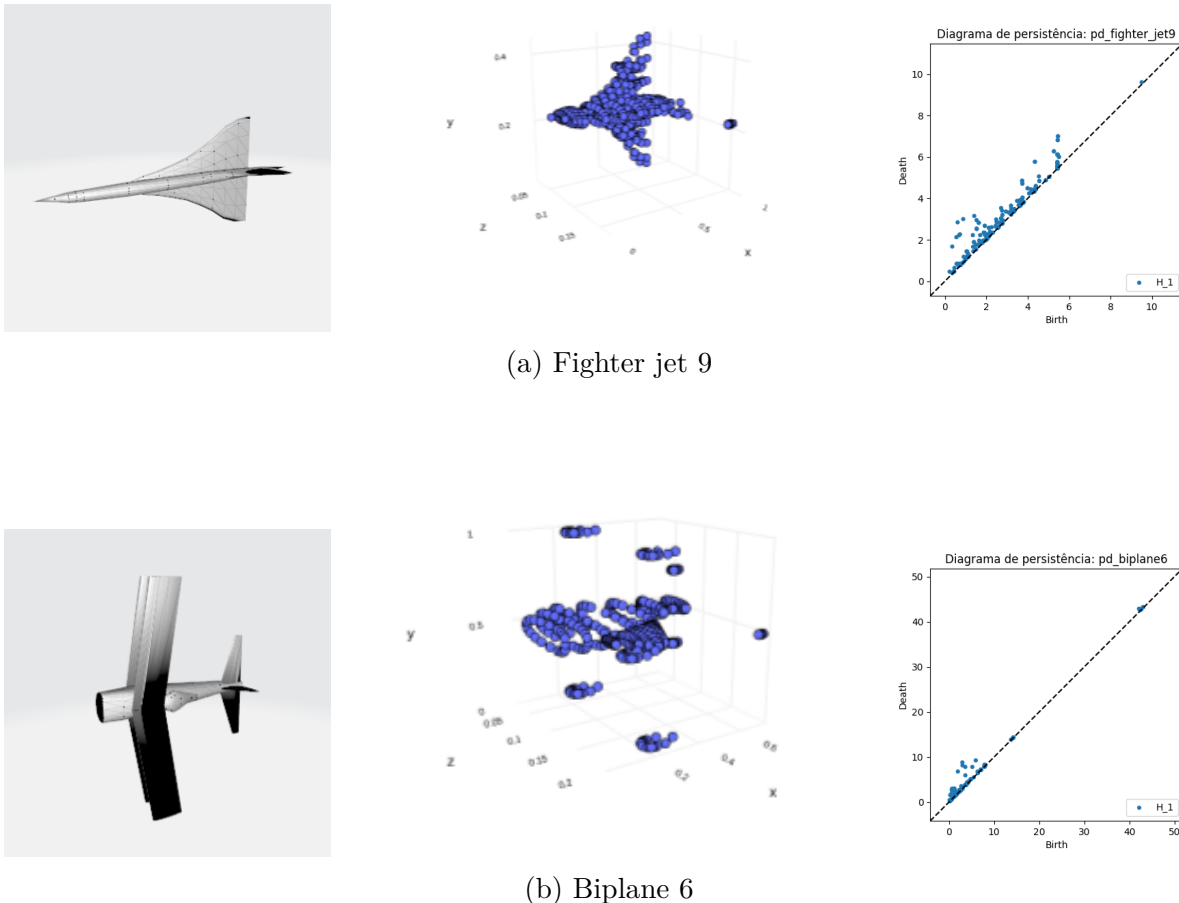


Figura 4.14: Imagem próxima à Fighter jet 9.

Além da nuvem de pontos, a Figura 4.14 destaca a imagem original e o diagrama de persistência, visto que os cálculos da distância utilizam a informação deste. A seguir, encontra-se o programa 4.12 criado para o cálculo e exibição do gráfico referente a distância entre duas imagens.

```

1 def bottle(caminho_pasta_pd, nome_arquivo1, nome_arquivo2, salvar=False,
2           caminho_salvar=''):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     arquivo1 = np.loadtxt(pasta_pd+'/pd_'+nome_arquivo1+'.csv',
9                          delimiter=',')
10    arquivo2 = np.loadtxt(pasta_pd+'/pd_'+nome_arquivo2+'.csv',
11                          delimiter=',')
12
13    distancia_bottleneck, matching = persim.bottleneck(arquivo1, arquivo2,
14                                                       matching=True)
15    persim.bottleneck_matching(arquivo1, arquivo2, matching,
16                              labels=['{} $H_1$'.format(nome_arquivo1),
17                                      '{} $H_1$'.format(nome_arquivo2)])
18    plt.title("Distância {:.4f}".format(distancia_bottleneck))
19    if salvar:
20        plt.savefig(caminho_salvar+'nome arquivo'+'.png')
21    plt.show()

```

Programa 4.12: Função para calcular a distância Bottleneck entre dois diagramas de persistência e gerar o gráfico.

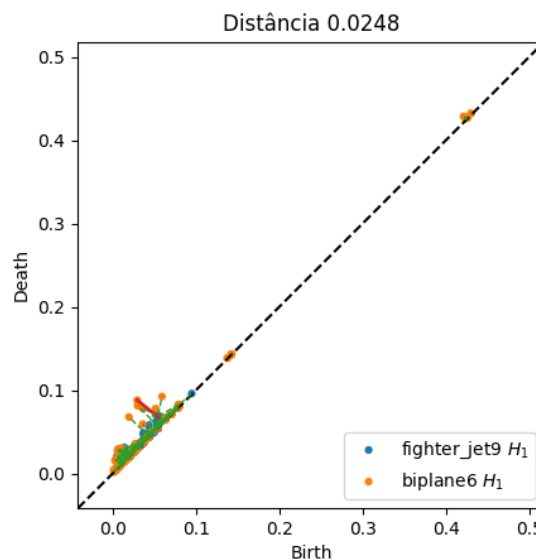


Figura 4.15: Distância Bottleneck entre Fighter Jet 9 e Biplane 6.

4.5 Distância entre as Imagens

As imagens de persistência são representações vetoriais dos pares – (nascimento, persistência) – de um complexo simplicial filtrado.

A informação matricial de uma imagem de persistência pode ser representada de forma pseudo-colorida, de acordo com os parâmetros escolhidos, como vimos na seção 4.3.

Estas matrizes possibilitam a comparação entre as imagens de forma simplificada e computacionalmente rápida. Basta medir a distância entre as respectivas entradas das matrizes por meio de uma métrica. Para este trabalho, as comparações foram feitas em três métricas⁶: Euclidiana, do Máximo e da Soma, utilizando a biblioteca `scipy.spatial.distance` do Python, que calcula estas distâncias de forma automática.

Observando as Figuras 4.6, 4.7, 4.8 e 4.9, imagem (a), é possível notar que o tamanho dos eixos de cada gráfico diferem. Isto acontece devido à filtração de cada imagem, ou seja, ao tamanho do nascimento e persistência de cada simplexo.

Então, para que a comparação seja possível, é preciso que as imagens tenham um tamanho padrão e, por consequência, uma matriz de mesma dimensão. Para isso, verificou-se os intervalos de nascimento e persistência de cada imagem e foi tomado o maior deles, como descreve o Programa 4.13.

```

1  pasta_pd = '...' # caminho da pasta dos PDs
2
3  maior_nascimento = 0
4  menor_nascimento = 100
5
6  maior_persistencia = 0
7  menor_persistencia = 100
8
9  for i in os.listdir(pasta_pd):
10     pd = np.loadtxt(pasta_pd+i, delimiter=',')
11     pd = pd*10
12
13     maior_n = max(pd[:, 0])
14     menor_n = min(pd[:, 0])
15
16     maior_p = max(pd[:, 1]-pd[:, 0])
17     menor_p = min(pd[:, 1]-pd[:, 0])
18
19     if maior_n > maior_nascimento:
20         maior_nascimento = maior_n
21
22     if menor_n < menor_nascimento:
23         menor_nascimento = menor_n
24
25     if maior_p > maior_persistencia:
26         maior_persistencia = maior_p
27
28     if menor_p < menor_persistencia:
29         menor_persistencia = menor_p
30
31 print('O menor nascimento é %.4f e o maior nascimento é %.4f' %
32       (menor_nascimento, maior_nascimento))
33 print('A menor persistência é %.4f e a maior persistência é %.4f' %

```

⁶Disponível em: <https://eu-provo.blogspot.com/2022/06/metrica.html>.

34

(menor_persistencia, maior_persistencia))

Programa 4.13: Encontrar tamanho ideal para intervalo de nascimento e persistência.

```
O menor nascimento é 0.0000 e o maior nascimento é 20.0000
A menor persistência é 0.0000 e a maior persistência é 8.2843
```

Agora, com a informação de intervalo ideal, foram escolhidos os intervalo de nascimento $[0, 21)$ e persistência $[0, 9)$ e, assim, novas imagens de persistência foram geradas. Proporcionando matrizes de tamanho 210×90 .

Para criar as novas imagens, foi utilizado o Programa 4.5, alterando $a=a*10$ (para gerar uma matriz menor), excluindo o valor maior e mudando os intervalos dos eixos: $\text{birth_range}=(0, 21)$ e $\text{pers_range}=(0, 9)$.

Exemplo 4.5. Imagem de persistência da figura Table 0 gerada com o novo intervalo.

```
# matriz Imagem de Persistência com peso, Table 0

[[4.43687043e-05  4.55904777e-05  4.59203323e-05 ...  4.92015251e-41
  4.92015251e-41  4.92015251e-41]
 [5.07785578e-05  5.21780197e-05  5.25566964e-05 ... -7.02544954e-27
 -7.02544954e-27 -7.02544954e-27]
 [5.69705388e-05  5.85418491e-05  5.89678858e-05 ...  3.93613845e-40
  3.93613845e-40  3.93613845e-40]
 ...
 [0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
 [0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
  0.00000000e+00  0.00000000e+00]
 [0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  0.00000000e+00
  0.00000000e+00  0.00000000e+00]]

(210, 90) # dimensão da matriz (colunas, linhas)
```

Para fins de comparação, também foram geradas informações com o Programa 4.6 com as mesmas alterações anteriores, mantendo apenas a característica de que estas, serão imagens padrão, sem alteração do peso.

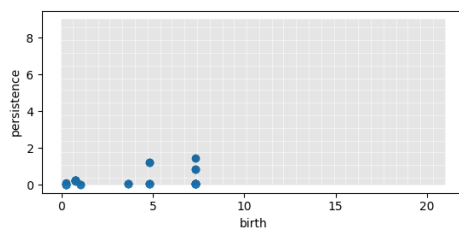
```
# matriz Imagem de Persistência, Table 0

[[2.58341637e-03  2.61237989e-03  2.61550571e-03 ...  2.12176010e-20
  1.00305745e-20  4.62959311e-21]
 [2.75012925e-03  2.78117233e-03  2.78470842e-03 ...  3.42070664e-20
  1.56886907e-20  7.19962957e-21]
 [2.89938850e-03  2.93232708e-03  2.93626392e-03 ...  5.40048043e-20
```

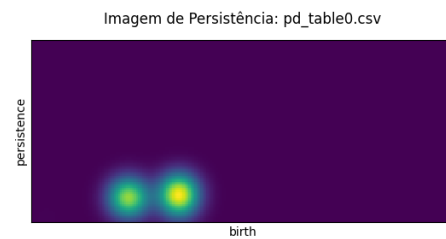
```

2.57180156e-20 1.13187536e-20]
...
[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
0.00000000e+00 0.00000000e+00]
[0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00
0.00000000e+00 0.00000000e+00]]
(210, 90)

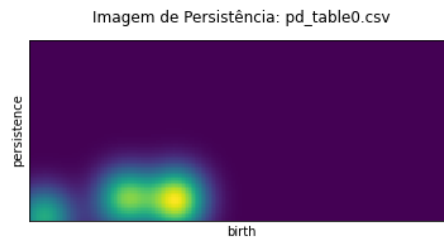
```



(a) PD (n,p)



(b) PI com peso



(c) PI padrão

Figura 4.16: Table 0, 60 pontos.

Exemplo 4.6. Imagem de persistência do Cubo 4.1c após ajuste de intervalos.

```

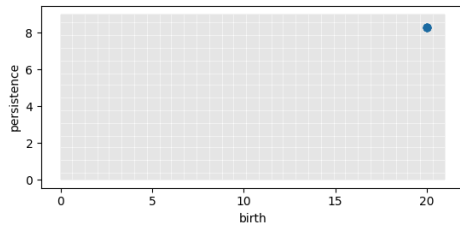
# matriz Imagem de Persistência com peso, Untitled 3

[[7.62851927e-201 3.91005281e-200 1.96449915e-199 ... 1.53806986e-171
1.38757027e-171 1.22705062e-171]
[4.06076038e-199 2.08137215e-198 1.04572854e-197 ... 8.18734663e-170
7.38621766e-170 6.53175059e-170]
[2.11883045e-197 1.08602189e-196 5.45642015e-196 ... 4.27200763e-168
3.85399320e-168 3.40814792e-168]
...
[1.70471125e-028 8.73762097e-028 4.38997882e-027 ... 3.43705626e+001
3.10074152e+001 2.74203540e+001]

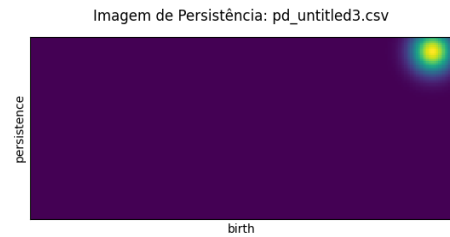
```

```
[1.45304610e-028 7.44769305e-028 3.74188980e-027 ... 2.92964642e+001
 2.64298156e+001 2.33723093e+001]
[1.21404985e-028 6.22270043e-028 3.12642574e-027 ... 2.44777972e+001
 2.20826535e+001 1.95280442e+001]]
```

(210, 90)



(a) PD (n,p)



(b) PI com peso



(c) PI padrão

Figura 4.17: Cubo - untitled3, 8 pontos.

Depois de todos os ajustes nas imagens, as distâncias entre as matrizes com informações das Imagens de Persistência foram calculadas, gerando seis arquivos diferentes: Distância Euclidiana para imagens com e sem peso, Distância do Máximo para imagens com e sem peso e Distância da Soma para imagens com e sem peso.

```
1 def distancia(caminho_arquivo, distância='euclidiana', salvar=False,
2               caminho_salvar='', nome_arquivo=''):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     # filtrar arquivos da pasta
9
10    arquivos = []
11
12    for i in os.listdir(caminho_arquivo):
13        if i.endswith('.csv'):
```

```
14         arquivos.append(i)
15
16     n = len(arquivos)
17
18     # criar matrizes para distâncias e tempos
19
20     matriz_distancia = np.zeros((n, n))
21     matriz_tempo = np.zeros((n, n))
22
23     for i in range(n):
24         for j in range(i+1, n):
25
26             tempo_inicial = time.process_time()
27
28             print('*Distância de %s até %s' % (arquivos[i], arquivos[j]))
29             ↪ # mensagem criada para acompanhar o processo
30
31             # ler arquivo
32             a1 = np.loadtxt(caminho_arquivo+'/'+arquivos[i],
33                             delimiter=',')
34             b1 = np.loadtxt(caminho_arquivo+'/'+arquivos[j],
35                             delimiter=',')
36
37             # vetorizar a matriz
38             a2 = a1.flatten()
39             b2 = b1.flatten()
40
41             # calcular as distâncias
42             if distância == 'euclidiana':
43                 distancia = ssd.euclidean(a2, b2)
44             if distância == 'máximo':
45                 distancia = ssd.chebyshev(a2, b2)
46             if distância == 'soma':
47                 distancia = ssd.cityblock(a2, b2)
48
49             # guardar as informações nas matrizes
50             tempo_final = time.process_time()
51
52             matriz_distancia[i, j] = distancia
53             matriz_tempo[i, j] = tempo_final-tempo_inicial
54
55             print('Distância %s: %.4f, tempo de execução: %.4f segundos.'
56                   % (distância, matriz_distancia[i, j],
57                      matriz_tempo[i, j]))
58
59             # salvar informações
60
61             if salvar:
62
63                 # salvar as distâncias em um csv
```

```

63     arquivo_salvar = open(caminho_salvar+'nome_arquivo.csv', 'a')
        ↳ # (nome do arquivo a ser criado/aberto, r=read, a=append)
64     np.savetxt(arquivo_salvar, matriz_distancia[i], newline=',')
        ↳ # (onde salvar, o que acrescentar, como separar a próxima
        ↳ informação - se não colocar nada vai pra próxima linha)
65     arquivo_salvar.write("\n") # encerra a linha que foi
        ↳ acrescentada e cria uma nova
66     arquivo_salvar.close() # fecha o arquivo
67
68     # salvar o tempo de execução em um csv
69     arquivo_tempo = open(caminho_salvar + 'nome_arquivo.csv', 'a')
70     np.savetxt(arquivo_tempo, matriz_tempo[i], fmt='%.4f',
71               newline=',') # fmt: qual o formato que você deseja
        ↳ salvar, neste caso, um número com 4 casas
        ↳ decimais
72     arquivo_tempo.write("\n")
73     arquivo_tempo.close()
74
75     return matriz_distancia

```

Programa 4.14: Função para calcular distância entre as matrizes das imagens de persistência.

Este programa 4.14 é único para as três métricas, sendo suficiente a alteração da opção *distância*.

As matrizes geradas, são apenas numéricas, superiores e possuem a última coluna vazia, pela forma como foram construídas e gravadas – por linhas. Objetivando torná-las mais claras, além da exclusão da coluna vazia; o nome dos arquivos foram inseridos como índices para linhas e colunas, por meio dos programas 4.8 e 4.9. Desta forma, ao abrir o arquivo gerado, é possível encontrar a distância entre duas imagens, procurando pela linha e coluna correspondentes.

4.5.1 Distância Euclidiana

A métrica Euclidiana é a mais conhecida dentre as três, utilizada desde o ensino básico. Esta métrica encontra o caminho mais curto entre dois pontos, e seu algoritmo é obtido pelo Teorema de Pitágoras.

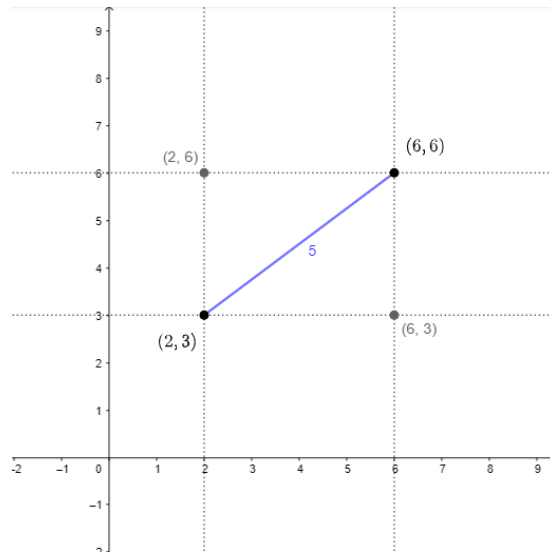
Definição 4.7 (Distância Euclidiana). Considere o espaço euclidiano $\mathbb{R}^n$. Sejam as n -uplas $p = (x_1, \dots, x_n)$ e $q = (y_1, \dots, y_n)$, que são pontos deste espaço.

Temos:

$$D_{\text{euclidiana}}(p, q) = \sqrt{(x_1 - y_1)^2 + \dots + (x_n - y_n)^2}$$

Exemplo 4.8. Considere os pontos $A = (2, 3)$ e $B = (6, 6)$ do plano. Temos que a distância euclidiana entre eles é dada por:

$$D_{\text{euclidiana}}(A, B) = \sqrt{(2 - 6)^2 + (3 - 6)^2} = \sqrt{16 + 9} = \sqrt{25} = 5$$



As Figuras 4.18 e 4.19 são as matrizes de distância euclidiana resultantes, geradas pelos Programas 4.14, 4.8 e 4.9. Para calcular a Distância Euclidiana, foram necessários 16,49 minutos e para a Distância Euclidiana para as imagens com peso, 15,71 minutos.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_ch
2	biplane0.ply	0	0.2958752997	0.1143497601	0.096402303	0.1128420176	0.09393325896	0.1161098068	0.1114358262	0.1963524675	0.1
3	biplane3.ply	0.2958752997	0	0.2316449035	0.2445939053	0.2386935093	0.2852401197	0.3406661901	0.228831928	0.1249252033	0.1
4	biplane4.ply	0.1143497601	0.2316449035	0	0.03278848765	0.01412738958	0.05745920528	0.1102932001	0.02041028915	0.1111045455	0.0
5	biplane6.ply	0.096402303	0.2445939053	0.03278848765	0	0.04034648674	0.04377786737	0.1002530186	0.02247202437	0.1305818211	0.1
6	biplane7.ply	0.1128420176	0.2386935093	0.01412738958	0.04034648674	0	0.05470190822	0.1042885066	0.02995160475	0.1169733419	0.1
7	biplane8.ply	0.09393325896	0.2852401197	0.05745920528	0.04377786737	0.05470190822	0	0.0567093291	0.05718822514	0.1670444637	0.1
8	biplane9.ply	0.1161098068	0.3406661901	0.1102932001	0.1002530186	0.1042885066	0.0567093291	0	0.1130194245	0.2203835447	0.1
9	desk_chair1.ply	0.1114358262	0.228831928	0.02041028915	0.02247202437	0.02995160475	0.05718822514	0.1130194245	0	0.1116205224	0.0
10	desk_chair2.ply	0.1963524675	0.1249252033	0.1111045455	0.1305818211	0.1169733419	0.1670444637	0.2203835447	0.1116205224	0	0.0
11	desk_chair5.ply	0.1721970191	0.1542813223	0.08136438933	0.1031568894	0.0869114718	0.1378952579	0.1906242682	0.08363608704	0.03040958693	
12	desk_chair6.ply	0.2178176318	0.1015375193	0.1361606527	0.1549291112	0.1417974479	0.1918187277	0.2452160488	0.1361689731	0.02531679734	0.0
13	desk_chair7.ply	0.1842845294	0.1378122069	0.09752107815	0.1117689139	0.1031209555	0.1530619854	0.20634322	0.09787072421	0.01491116399	0.0
14	desk_chair8.ply	0.3344076281	0.441286787	0.3281581152	0.3265121807	0.3268283565	0.320655558	0.3219725625	0.3289084187	0.368791395	0.1
15	dining_chair0.ply	0.1833350695	0.1405754811	0.09589481212	0.1162953693	0.1009912919	0.1514263907	0.2043021012	0.09675294252	0.01790581262	0.0
16	dining_chair1.ply	0.5038872261	0.2194111207	0.4396928958	0.4540948429	0.4472781865	0.4948908714	0.5496652988	0.4385194873	0.3324000471	0
17	dining_chair2.ply	0.1310979318	0.2062892024	0.02728200857	0.05216475476	0.03362230869	0.08285708158	0.1357629879	0.03179258521	0.08469664293	0.0
18	dining_chair3.ply	0.1556885911	0.1759388548	0.06103584433	0.08400901986	0.06514655387	0.1160852068	0.1681795017	0.06355930288	0.05323662842	0.0
19	dining_chair4.ply	0.1576584982	0.1686175943	0.06510631444	0.0844912082	0.07288197281	0.1209911802	0.1748925374	0.06577629031	0.04789824765	0.0
20	dining_chair5.ply	0.3242441964	0.2941067482	0.3360540091	0.3194920857	0.3480434524	0.3546923147	0.3990241585	0.3248647977	0.3155362778	0.1
21	dining_chair6.ply	0.1917982423	0.1287955669	0.1065549497	0.1256649102	0.1123573951	0.1621055959	0.215568242	0.1065946828	0.006513657034	0.0

Figura 4.18: Recorte da matriz de distância euclidiana.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_chair3.ply
2	biplane0.ply	0	0.02528791555	0.009129757201	0.009523770223	0.006521692796	0.007013178211	0.1513631324	0.00740612684	0.02016510691	0.00
3	biplane3.ply	0.02528791555	0	0.01843449141	0.01839021418	0.0249458986	0.0257962464	0.1550867449	0.02700670857	0.02781958099	0.0
4	biplane4.ply	0.009129757201	0.01843449141	0	0.007145579546	0.01010088176	0.0110911673	0.1530680196	0.01214657931	0.01839350067	0.0
5	biplane6.ply	0.009523770223	0.01839021418	0.007145579546	0	0.007003215574	0.007825817124	0.1533601371	0.009142606142	0.02424939891	0.00
6	biplane7.ply	0.006521692796	0.0249458986	0.01010088176	0.007003215574	0	0.00101125814	0.1531407462	0.002248780727	0.02544981378	0.00
7	biplane8.ply	0.007013178211	0.0257962464	0.0110911673	0.007825817124	0.00101125814	0	0.1531913442	0.001337663502	0.02618885117	0.0
8	biplane9.ply	0.1513631324	0.1550867449	0.1530680196	0.1533601371	0.1531407462	0.1531913442	0	0.153202922	0.1504201078	0.
9	desk_chair1.ply	0.00740612684	0.02700670857	0.01214657931	0.009142606142	0.002248780727	0.001337663502	0.153202922	0	0.02674998447	0.00
10	desk_chair2.ply	0.02016510691	0.02781958099	0.01839350067	0.02424939891	0.02544981378	0.02618885117	0.1504201078	0.02674998447	0	0.0
11	desk_chair3.ply	0.005564913024	0.02586104698	0.01050236289	0.008072018589	0.001598686494	0.00168392931	0.1524984618	0.002145755607	0.02498119373	
12	desk_chair6.ply	0.05793807944	0.05707537839	0.05503279814	0.06109805572	0.06303027942	0.06377664384	0.1533405127	0.06431763607	0.03792027066	0.0
13	desk_chair7.ply	0.09277323176	0.09915978914	0.09564624985	0.09707071059	0.09681099361	0.09698187063	0.0891000253	0.09703419254	0.0874904737	0.0
14	desk_chair8.ply	1.06085413	1.060858514	1.060598473	1.062151779	1.06236112	1.062481434	1.030392751	1.06253046	1.053766855	
15	dining_chair0.ply	0.08637826413	0.09109016327	0.08802670475	0.08799616054	0.0876963611	0.08773925147	0.07429037346	0.08776628018	0.08800603118	0.0
16	dining_chair1.ply	0.06841295569	0.07161714635	0.06875621673	0.07282098027	0.07379833259	0.07429201457	0.1405433506	0.07462412558	0.05254914672	0.0
17	dining_chair2.ply	0.02205747639	0.03557497475	0.02595255321	0.02564234699	0.02416554312	0.02422948078	0.1357792149	0.02423733154	0.03068652093	0.0
18	dining_chair3.ply	0.1187589799	0.1245642616	0.1217608063	0.1227147848	0.1224311084	0.1225623929	0.08886345296	0.1225898856	0.1154675066	0.
19	dining_chair4.ply	0.007502840333	0.02503090967	0.01069669614	0.00795682528	0.004523651193	0.004737600185	0.1509758881	0.005285129884	0.02550616121	0.00
20	dining_chair5.ply	0.007315795974	0.02761446165	0.01362839975	0.01227046696	0.008963876167	0.009017380049	0.1472826959	0.009046913434	0.02428205103	0.00

Figura 4.19: Recorte da matriz de distância euclidiana das imagens com peso.

Como já dito, é possível conhecer a distância entre as imagens por observação, pela facilidade que os índices proporcionam. Nota-se também, que o parâmetro peso na criação das imagens influencia diretamente nas distâncias entre elas.

Porém, como a matriz gerada é muito grande, não é possível perceber quais imagens são mais próximas, isto é, mais parecidas segundo os cálculos realizados. Assim, utilizou-se o Programa 4.10 para constatar e destacar quais se assemelham.

Para uma distância escolhida, o programa retorna um dicionário, como abaixo, e cria uma matriz que elenca o nome das imagens que apresentam tal proximidade, como em 4.18 e 4.19. Figuras que não estão dentro deste parâmetro, não são citadas.

```
# imagens (sem peso) com distância de até 0.01

{'biplane4.ply': ['ship6.ply'], 'desk_chair1.ply': ['human3.ply'],
 → 'desk_chair2.ply': ['dining_chair6.ply'], 'desk_chair7.ply':
 → ['dining_chair0.ply', 'dining_chair6.ply', 'fish3.ply',
 → 'handgun2.ply'], [...], 'ship3.ply': ['flying_bird5.ply'],
 → 'ship6.ply': ['biplane4.ply', 'flying_bird9.ply'], 'ship9.ply':
 → ['handgun7.ply', 'human4.ply', 'handgun8.ply']}
```

A diferença entre distâncias resultantes é grande, como podemos perceber nos trechos exibidos nas Figuras 4.18 e 4.19. Sendo assim, foram usados critérios diferentes nas filtragens das imagens com e sem peso para obter um melhor resultado.

	A	B	C	D	E
1		1	2	3	4
2	biplane4.ply	ship6.ply			
3	desk_chair1.ply	human3.ply			
4	desk_chair2.ply	dining_chair6.ply			
5	desk_chair7.ply	dining_chair0.ply	dining_chair6.ply	fish3.ply	handgun2.ply
6	dining_chair0.ply	desk_chair7.ply	fish3.ply		
7	dining_chair2.ply	sedan0.ply			
8	dining_chair3.ply	desk_chair3.ply			
9	dining_chair6.ply	desk_chair2.ply	desk_chair7.ply	handgun2.ply	
10	fighter_jet8.ply	helicopter9.ply	biplane5.ply		
11	fighter_jet0.ply	sedan0.ply	fish3.ply	fish4.ply	

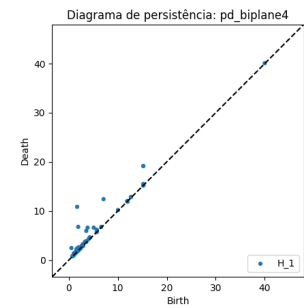
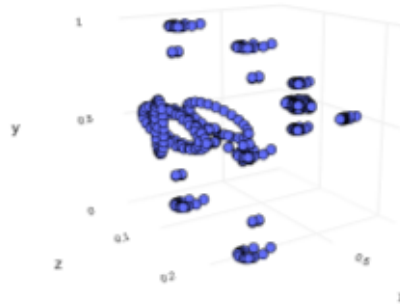
Figura 4.20: Matriz de Imagens padrão que possuem distância euclidiana menor do que 0,01.

	A	B	C	D	E	F	G
1		1	2	3	4	5	6
2	desk_chair1.ply	flying_bird4.ply	sword0.ply	sword5.ply	sword6.ply	sword7.ply	sword8.ply
3	dining_chair9.ply	human1.ply					
4	fish5.ply	ship0.ply	ship5.ply	helicopter0.ply			
5	fish8.ply	guitar8.ply	ship2.ply				
6	flying_bird0.ply	human3.ply					
7	flying_bird4.ply	desk_chair1.ply	sword0.ply	sword7.ply	sword8.ply		
8	guitar8.ply	fish8.ply	ship0.ply	ship2.ply	sword1.ply		
9	helicopter8.ply	ship8.ply					
10	human1.ply	dining_chair9.ply					
11	ship8.ply	fish5.ply	guitar8.ply	ship2.ply	ship5.ply	sword4.ply	

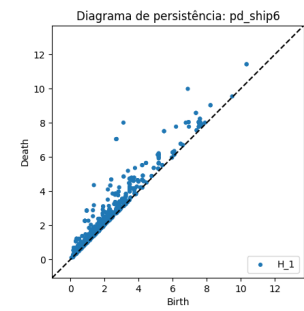
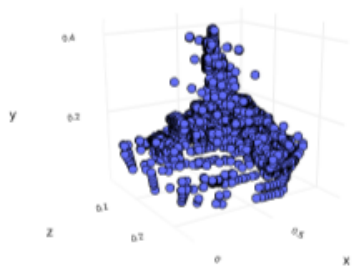
Figura 4.21: Matriz de Imagens com peso que possuem distância euclidiana menor do que 0,0001.

Os filtros gerados causam um certo estranhamento ao entender como próximas, figuras diferentes, como por exemplo **biplane 4** e **ship 6**, na segunda linha da Figura 4.20. Para observação, foi construído um código para gerar a nuvem dos pontos das figuras classificadas semelhantes.

As figuras abaixo mostram as imagens originais em 3D, sua nuvem de pontos e o Diagrama de Persistência, para comparação ocular.

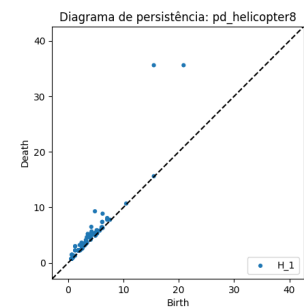
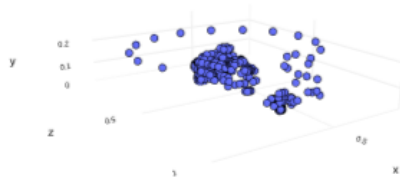


(a) Biplane 4

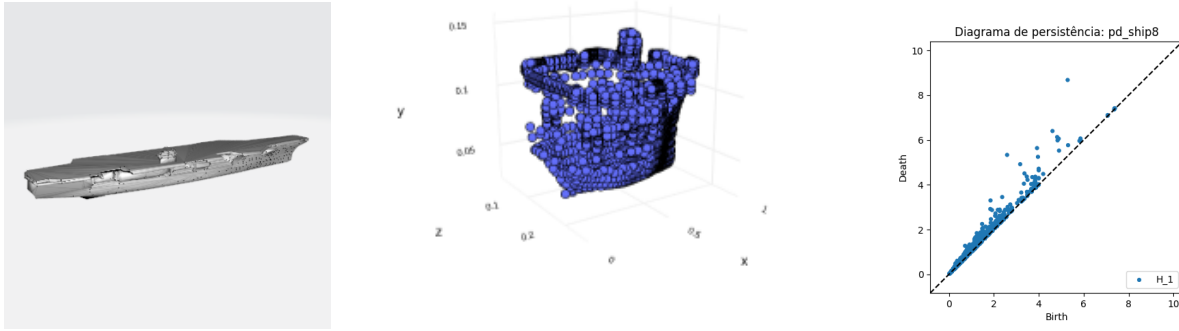


(b) Ship 6

Figura 4.23: Imagem próxima à Biplane 4 - linha 2 da Figura 4.20.



(a) Helicopter 8



(b) Ship 8

Figura 4.25: Imagem próxima à Helicopter 8 - linha 9 da Figura 4.21.

É importante observar que a semelhança entre as imagens, é topológica e pode ser melhor entendida ao observar os diagramas de persistência, que foram acrescentados com este objetivo particular.

A seguir, serão feitas as mesmas comparações que nesta subseção, porém com as métricas do máximo e da soma.

4.5.2 Distância do Máximo

A métrica do máximo, também conhecida como métrica de Chebyshév ou do tabuleiro de xadrez, é uma das principais métricas da topologia e é definida como segue.

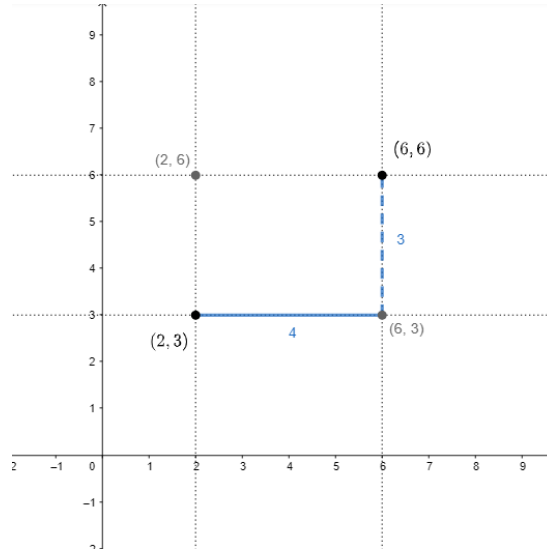
Definição 4.9 (Distância do Máximo). Considere o espaço euclidiano $\mathbb{R}^n$. Sejam as n -uplas $p = (x_1, \dots, x_n)$ e $q = (y_1, \dots, y_n)$, que são pontos deste espaço.

Temos:

$$D_{\text{máximo}}(p, q) = \max \{ |x_1 - y_1|, \dots, |x_n - y_n| \}.$$

Exemplo 4.10. Considere os pontos $A = (2, 3)$ e $B = (6, 6)$ do plano. Temos que a distância do máximo entre eles é dada por:

$$D_{\text{máximo}}(A, B) = \max \{ |2 - 6|, |3 - 6| \} = \max \{ 4, 3 \} = 4.$$



Para as imagens de persistência com e sem pesos, foram geradas as seguintes matrizes de distância. Sendo que levaram 17 minutos para as distâncias entre as imagens padrão 4.26 e 16,25 minutos para o cálculo entre as imagens com peso 4.27.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_chair5.ply
2	biplane0.ply	0	0.02502633152	0.006290908145	0.005797726568	0.00614965756	0.006082063317	0.006487720512	0.006175505122	0.01511823275	0.0
3	biplane3.ply	0.02502633152	0	0.02024233911	0.02232715293	0.01994377271	0.02475122591	0.02857980487	0.0204391805	0.01009486284	0.0
4	biplane4.ply	0.006290908145	0.02024233911	0	0.0027262525	0.0009463980764	0.004854587565	0.008530713689	0.001806616239	0.01026783083	0.00
5	biplane6.ply	0.005797726568	0.02232715293	0.0027262525	0	0.003203885782	0.002754123419	0.006826476703	0.002107008504	0.01229872601	0.00
6	biplane7.ply	0.00614965756	0.01994377271	0.0009463980764	0.003203885782	0	0.005347178842	0.008891308647	0.002585729391	0.01005385792	0.0
7	biplane8.ply	0.006082063317	0.02475122591	0.004854587565	0.002754123419	0.005347178842	0	0.00411819623	0.004735384151	0.01475045512	0.
8	biplane9.ply	0.006487720512	0.02857980487	0.008530713689	0.006826476703	0.008891308647	0.00411819623	0	0.008853580381	0.01868297372	0.0
9	desk_chair1.ply	0.006175505122	0.0204391805	0.001806616239	0.002107008504	0.002585729391	0.004735384151	0.008853580381	0	0.01039681627	0.00
10	desk_chair2.ply	0.01511823275	0.01009486284	0.01026783083	0.01229872601	0.01005385792	0.01475045512	0.01868297372	0.01039681627	0	0.00
11	desk_chair5.ply	0.01246060363	0.01269744351	0.007544895599	0.009781529804	0.00727711318	0.0121736898	0.01601066381	0.008038660194	0.002819305878	
12	desk_chair6.ply	0.01753921817	0.007626421994	0.01269211718	0.01475312678	0.01242604883	0.01720485588	0.02110395913	0.01286381111	0.002468440849	0.00
13	desk_chair7.ply	0.01421839779	0.01089187686	0.009395134383	0.01145164398	0.009147837488	0.01389894227	0.01778313676	0.009610009266	0.001019380247	0.00
14	desk_chair8.ply	0.01400052561	0.02594443455	0.0140015928	0.01400159452	0.01400145117	0.01400159463	0.01400159478	0.01400159474	0.01592635593	0.0
15	dining_chair0.ply	0.01438547923	0.01067203571	0.009570303393	0.01171251186	0.009271736992	0.01410348908	0.01793517424	0.009906224575	0.001187622928	0.00
16	dining_chair1.ply	0.03723878704	0.01500991057	0.032382659	0.03410737087	0.03235409742	0.0366966769	0.04068441263	0.03215177151	0.02289764436	0.0
17	dining_chair2.ply	0.00769574541	0.01746638136	0.002838731866	0.004985486014	0.002809875357	0.007370653828	0.01124797932	0.003379843447	0.00744787935	0.00
18	dining_chair3.ply	0.01150395528	0.01355880345	0.006710739851	0.008907995798	0.00638496926	0.01125104177	0.01505242388	0.007209367442	0.003873862029	0.00
19	dining_chair4.ply	0.01011814931	0.01570044948	0.005279258509	0.007118669298	0.005440329668	0.0096433824	0.01360926325	0.005314955995	0.005605586637	0.0
20	dining_chair5.ply	0.02155736753	0.02348912309	0.02196172487	0.02044910809	0.02282376996	0.02245442938	0.02475107196	0.02105057307	0.02055085558	0.0
21	dining_chair6.ply	0.01483288023	0.01036495808	0.009885268362	0.01201852458	0.009774865119	0.01447025368	0.0183976212	0.01012804411	0.0004320435238	0.00

Figura 4.26: Parte da matriz de distância do máximo.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_chair3.ply
2	biplane0.ply	0	0.002501954822	0.0007716367613	0.0008099945258	0.0005712445115	0.0005999464215	0.01214783974	0.0006058395432	0.001655689327	0.000
3	biplane3.ply	0.002501954822	0	0.002137483248	0.001809468617	0.002434301061	0.002486525478	0.01218443757	0.002590308053	0.00229212716	0.00
4	biplane4.ply	0.0007716367613	0.002137483248	0	0.0006233642185	0.0008446414642	0.0009245650847	0.01218331662	0.0009992430883	0.001591665535	0.000
5	biplane6.ply	0.0008099945258	0.001809468617	0.0006233642185	0	0.0007444944359	0.0007927951938	0.01218529056	0.0009240060822	0.002024923818	0.000
6	biplane7.ply	0.0005712445115	0.002434301061	0.0008446414642	0.0007444944359	0	8.69E-05	0.01218485436	0.0002038049459	0.002058295737	0.00
7	biplane8.ply	0.0005999464215	0.002486525478	0.0009245650847	0.0007927951938	8.69E-05	0	0.01218531498	0.0001411799479	0.002099979188	0.000
8	biplane9.ply	0.01214783974	0.01218443757	0.01218331662	0.01218529056	0.01218485436	0.01218531498	0	0.01218531774	0.01212018327	0.0
9	desk_chair1.ply	0.0006058395432	0.002590308053	0.0009992430883	0.0009240060822	0.0002038049459	0.0001411799479	0.01218531774	0	0.002116067549	0.000
10	desk_chair2.ply	0.001655689327	0.00229212716	0.001591665535	0.002024923818	0.002058295737	0.002099979188	0.01212018327	0.002116067549	0	0.0
11	desk_chair5.ply	0.0004882404252	0.002534772613	0.0008831734783	0.0008517916867	0.000110345794	0.0001227692729	0.01216130357	0.0001485629237	0.0020067389	0.000
12	desk_chair6.ply	0.004651594369	0.004661004935	0.004466817352	0.004964510154	0.005007064246	0.00504626267	0.01194702289	0.005064826685	0.003017035286	0.00
13	desk_chair7.ply	0.007357936238	0.007540964878	0.007525480178	0.007553805465	0.007550305956	0.00755431734	0.00760596891	0.007554417333	0.007155242539	0.00
14	desk_chair8.ply	0.08450396963	0.08452286462	0.08450549463	0.08453403125	0.08453445474	0.08453542387	0.0836540651	0.08453556559	0.08426563018	0.0
15	dining_chair0.ply	0.006905222348	0.00693732581	0.006936517559	0.006937909639	0.006937499648	0.00693795349	0.0063239752	0.006937955876	0.006896851165	0.00
16	dining_chair1.ply	0.005449224276	0.005757245823	0.005643289466	0.005840266357	0.005838528958	0.005852493558	0.01133852245	0.005854601507	0.004398997546	0.00
17	dining_chair2.ply	0.001871456982	0.002573525368	0.001937472161	0.001942249456	0.00194065907	0.001942342089	0.01122117991	0.001942353202	0.001904836424	0.00
18	dining_chair3.ply	0.009751958495	0.009924482369	0.009915870768	0.00993392728	0.009930147201	0.009934176061	0.007213588177	0.009934233566	0.00966675784	0.00
19	dining_chair4.ply	0.0005714711772	0.002425586165	0.000816333339	0.0007054580356	0.00039914971	0.0003992903212	0.01210559882	0.0003992915751	0.002053088874	0.000
20	dining_chair5.ply	0.0005251628285	0.002548306547	0.0009516730505	0.0008692650214	0.0006745192086	0.0006798036664	0.01193045047	0.0006798789456	0.001969698356	0.000
21	dining_chair6.ply	0.157230408	0.1572321499	0.1572321174	0.1572321548	0.1572321497	0.1572321645	0.1508955254	0.1572321646	0.1572299927	0.0

Figura 4.27: Parte da matriz de distância do máximo das imagens com peso.

Utilizando o mesmo programa 4.10, com um parâmetro máximo de 0,001 para imagens padrão e de 0,00001 para imagens com peso, obtivemos os arquivos listados nas Figuras 4.28 e 4.29.

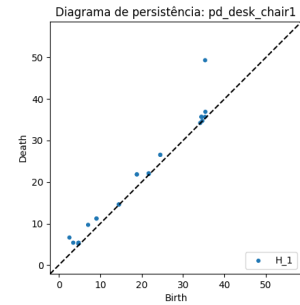
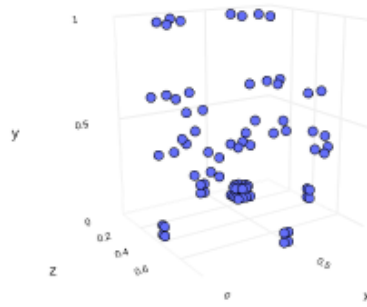
	A	B	C	D	E
1		1	2	3	4
2	biplane4.ply	biplane7.ply	flying_bird9.ply	ship6.ply	
3	biplane7.ply	biplane4.ply	flying_bird9.ply	ship6.ply	
4	desk_chair1.ply	human3.ply			
5	desk_chair2.ply	dining_chair6.ply			
6	desk_chair7.ply	dining_chair0.ply	dining_chair6.ply	fish3.ply	handgun2.ply
7	dining_chair0.ply	desk_chair7.ply	dining_chair6.ply	fish3.ply	handgun2.ply
8	dining_chair2.ply	sedan7.ply	sedan0.ply		
9	dining_chair6.ply	desk_chair2.ply	desk_chair7.ply	dining_chair0.ply	handgun2.ply
10	fighter_jet8.ply	helicopter9.ply	biplane5.ply		
11	fighter_jet9.ply	fish2.ply	fish4.ply		

Figura 4.28: Matriz de Imagens padrão que possuem distância do máximo menor do que 0,001.

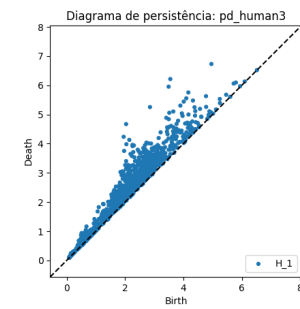
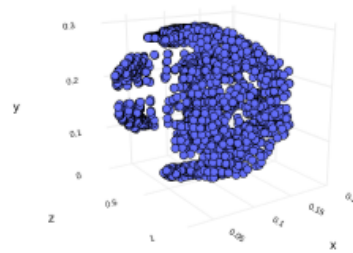
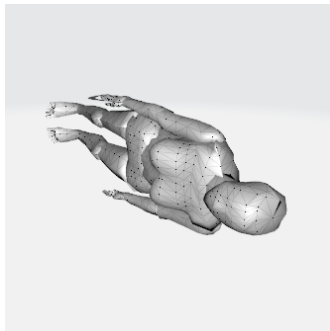
	A	B	C	D	E	F
1		1	2	3	4	5
2	desk_chair1.ply	flying_bird4.ply	sword0.ply	sword5.ply	sword6.ply	sword8.ply
3	dining_chair9.ply	human1.ply				
4	fish5.ply	ship5.ply				
5	fish8.ply	guitar8.ply	ship2.ply			
6	flying_bird4.ply	desk_chair1.ply	sword0.ply	sword8.ply		
7	guitar8.ply	fish8.ply	ship0.ply	ship2.ply		
8	helicopter8.ply	ship8.ply				
9	human1.ply	dining_chair9.ply				
10	ship0.ply	guitar8.ply	ship2.ply	ship5.ply	sword1.ply	
11	fish5.ply	fish8.ply	ship5.ply	ship8.ply		

Figura 4.29: Matriz de Imagens padrão que possuem distância do máximo menor do que 0,00001.

Dentre as comparações realizadas, as imagens próximas à **Desk chair 1** da linha 4 da Figura 4.28 e à **Fish 5** da linha 4 da Figura 4.29 foram geradas para observação.

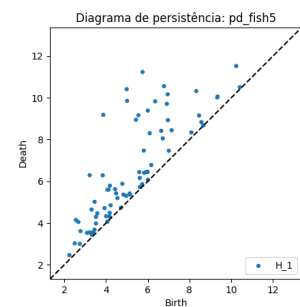
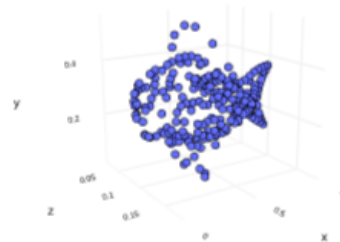
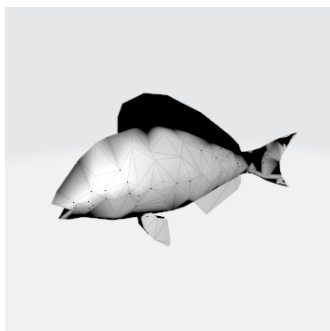


(a) Deskchair 1

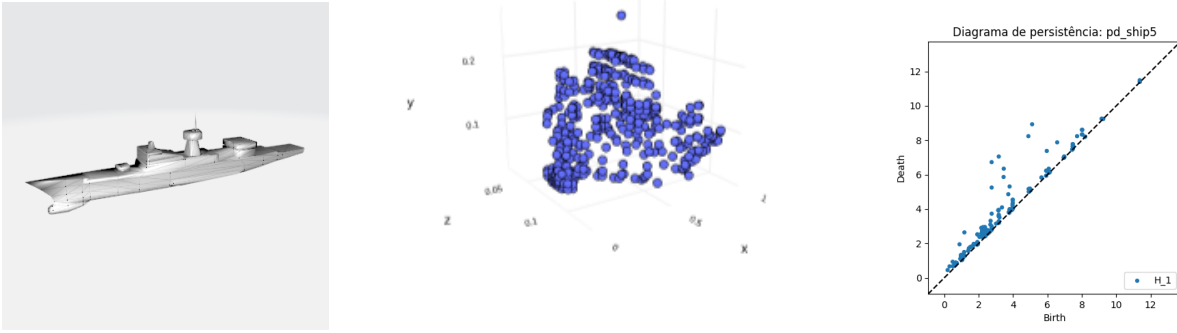


(b) Human 3

Figura 4.31: Imagem próxima à Deskchair 1 - linha 4 da Figura 4.28.



(a) Fish 5



(b) Ship 5

Figura 4.33: Imagem próxima à Fish 5 - linha 4 da Figura 4.29.

4.5.3 Distância da Soma

Por fim, a métrica da soma, também conhecida como métrica de Manhattan ou ainda, como métrica do táxi, tem como ideia medir a distância entre dois pontos “contornando quarteirões”.

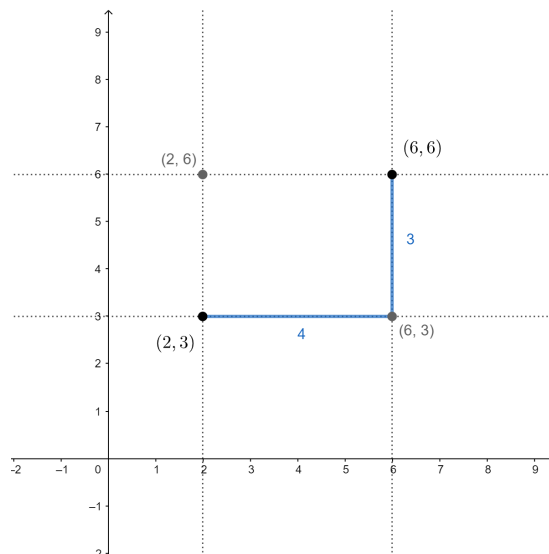
Definição 4.11 (Distância da Soma). Considere o espaço euclidiano $\mathbb{R}^n$. Sejam as n -uplas $p = (x_1, \dots, x_n)$ e $q = (y_1, \dots, y_n)$, que são pontos deste espaço.

Temos:

$$D_{\text{soma}}(p, q) = |x_1 - y_1| + \dots + |x_n - y_n|.$$

Exemplo 4.12. Considere os pontos $A = (2, 3)$ e $B = (6, 6)$ do plano. Temos que a distância da soma entre eles é dada por:

$$D_s(A, B) = |2 - 6| + |3 - 6| = 4 + 3 = 7.$$



Para esta métrica, foram produzidos os resultados a seguir. O tempo gasto para gerar a matriz de distância entre as imagens padrão 4.34 foi de 16,47 minutos e dentre as imagens com peso 4.35, um tempo de 15,81 minutos.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_chair5.ply
2	biplane0.ply	0	7.489490124	3.740585685	3.116978524	3.63829769	2.669292828	3.846851183	3.580867667	5.683776458	5.
3	biplane3.ply	7.489490124	0	5.142096276	5.106819638	5.450260109	6.251828432	7.632595771	5.096293214	3.054347513	3.
4	biplane4.ply	3.740585685	5.142096276	0	0.8464698164	0.4345203446	1.21571435	2.49120726	0.4832982217	2.120563449	1.
5	biplane6.ply	3.116978524	5.106819638	0.8464698164	0	1.053233501	1.145008911	2.52577625	0.553857374	2.569923931	2.
6	biplane7.ply	3.63829769	5.450260109	0.4345203446	1.053233501	0	1.126639523	2.259996141	0.7510911778	2.512900045	1.
7	biplane8.ply	2.669292828	6.251828432	1.21571435	1.145008911	1.126639523	0	1.380767339	1.188724016	3.317576279	2.
8	biplane9.ply	3.846851183	7.632595771	2.49120726	2.52577625	2.259996141	1.380767339	0	2.536302557	4.580861511	3.
9	desk_chair1.ply	3.580867667	5.096293214	0.4832982217	0.553857374	0.7510911778	1.188724016	2.536302557	0	2.153808933	1.
10	desk_chair2.ply	5.683776458	3.054347513	2.120563449	2.569923931	2.512900045	3.317576279	4.580861511	2.153808933	0	0.6
11	desk_chair5.ply	5.182344197	3.657010734	1.533821029	2.091063012	1.913381965	2.740467547	3.978146178	1.620408963	0.6033941244	
12	desk_chair6.ply	6.065494536	2.609508802	2.555302083	2.949158638	2.947860717	3.738674936	5.023087411	2.560382028	0.473568693	1.
13	desk_chair7.ply	5.361773179	3.400663645	1.779151148	2.253654631	2.160238587	2.96474782	4.231932126	1.806452904	0.3559576195	0.3
14	desk_chair8.ply	15.91247272	20.21141417	15.12758151	15.13424018	14.80534696	14.00689156	14.27296487	15.17150254	17.22878444	16.
15	dining_chair0.ply	5.319043952	3.49953004	1.701143869	2.221417342	2.066305452	2.886040353	4.132642767	1.747752764	0.4486895166	0.3
16	dining_chair1.ply	12.74900014	5.276388973	10.05694809	10.08110805	10.42686705	11.17620769	12.54677644	10.01118349	7.96591493	8.
17	dining_chair2.ply	4.177917788	4.774592874	0.4949279144	1.179629478	0.7970713997	1.649870186	2.858002897	0.66447214	1.722977043	1.
18	dining_chair3.ply	4.707107844	4.282766034	1.035643611	1.683935376	1.302195008	2.173033465	3.349829737	1.151468621	1.231031774	0.6
19	dining_chair4.ply	4.91590927	3.832644635	1.335278365	1.810342227	1.729320869	2.532749389	3.800208307	1.376816645	0.788122483	0.3
20	dining_chair5.ply	9.059347181	8.230629343	9.182509842	8.881528144	9.451468921	9.985968687	11.36673603	8.989488469	8.85798946	8.
21	dining_chair6.ply	5.532563879	3.192095389	1.978548007	2.4189187	2.385552953	3.162983884	4.4405000383	1.983856038	0.1718354457	0.5

Figura 4.34: Parte da matriz de distância da soma.

	A	B	C	D	E	F	G	H	I	J	
1		biplane0.ply	biplane3.ply	biplane4.ply	biplane6.ply	biplane7.ply	biplane8.ply	biplane9.ply	desk_chair1.ply	desk_chair2.ply	desk_chair5.ply
2	biplane0.ply	0	0.4874695626	0.2104110312	0.2145250658	0.1356594887	0.1462239232	3.675128437	0.1652122978	0.455879198	0.
3	biplane3.ply	0.4874695626	0	0.2971370519	0.3427116496	0.4498918995	0.4708141016	4.146361938	0.4917870043	0.6567279673	0.
4	biplane4.ply	0.2104110312	0.2971370519	0	0.1489678338	0.2119634765	0.2328932188	3.668224569	0.2538661215	0.399277255	0.
5	biplane6.ply	0.2145250658	0.3427116496	0.1489678338	0	0.1097167482	0.1281256902	3.863542463	0.1490985929	0.5446779883	0.
6	biplane7.ply	0.1356594887	0.4498918995	0.2119634765	0.1097167482	0	0.02092974234	3.766558429	0.04190264502	0.5783655459	0.
7	biplane8.ply	0.1462239232	0.4708141016	0.2328932188	0.1281256902	0.02092974234	0	3.766476036	0.02097290268	0.5992952883	0.
8	biplane9.ply	3.675128437	4.146361938	3.868224569	3.863542463	3.766558429	3.766476036	0	3.783127469	3.842362586	3
9	desk_chair1.ply	0.1652122978	0.4917870043	0.2538661215	0.1490985929	0.04190264502	0.02097290268	3.783127469	0	0.620268191	0.
10	desk_chair2.ply	0.6567279673	0.399277255	0.5446779883	0.5783655459	0.5992952883	3.842362586	0.620268191	0	0.	0.
11	desk_chair5.ply	0.1181294473	0.4677768721	0.2247468113	0.1452447679	0.04191452502	0.04226026464	3.731450071	0.05471444741	0.5736982251	
12	desk_chair6.ply	1.38151384	1.371968623	1.293744759	1.425614682	1.504823492	1.525753235	4.321912241	1.546726137	0.9264579464	
13	desk_chair7.ply	2.225130334	2.662410449	2.368564747	2.431026463	2.359008519	2.367406607	2.268964477	2.383382163	2.150642368	2
14	desk_chair8.ply	26.45740372	26.52315911	26.43995783	26.57015489	26.5835596	26.60121673	26.11270059	26.61634315	26.0889203	2
15	dining_chair0.ply	2.091411638	2.536156554	2.271545092	2.229380783	2.129800089	2.140636391	1.763201898	2.161604314	2.38078474	
16	dining_chair1.ply	1.702912871	1.779834322	1.614259048	1.720868801	1.826222524	1.847152266	3.893653223	1.868125169	1.247929093	1
17	dining_chair2.ply	0.502924679	0.9678477819	0.7024362682	0.6667720279	0.5650275678	0.5606956118	3.2255543	0.5582150095	0.8707163793	0.
18	dining_chair3.ply	2.659510463	3.129257323	2.837846359	2.872159577	2.785869462	2.78843946	2.278197033	2.797824269	2.688157054	2
19	dining_chair4.ply	0.1977258662	0.5300440559	0.2897355478	0.1930852031	0.09918013035	0.1168622517	3.684016389	0.1378351542	0.6325014155	0.
20	dining_chair5.ply	0.1971446655	0.6406315953	0.3814962822	0.3316690149	0.2270480143	0.2203794467	3.560833057	0.2225208851	0.6140646017	0.
21	dining_chair6.ply	49.5065815	49.86529505	49.69723947	49.68164104	49.5560586	49.54603986	45.51587741	49.53475225	49.73181329	4

Figura 4.35: Parte da matriz de distância da soma das imagens com peso.

Utilizando o mesmo programa 4.10, utilizando um parâmetro máximo de 0.1 para as imagens padrão e de 0.001 para imagens com peso, obtivemos os arquivos 4.36 e 4.37.

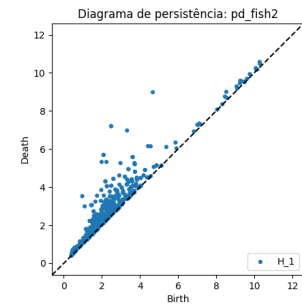
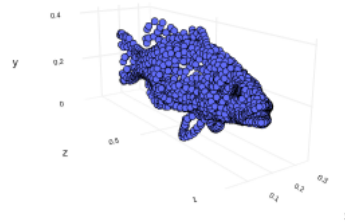
	A	B
1		1
2	fighter_jet8.ply	helicopter9.ply
3	fish2.ply	fish4.ply
4	fish4.ply	fish2.ply
5	helicopter9.ply	fighter_jet8.ply
6	human4.ply	handgun8.ply
7	handgun8.ply	human4.ply
8	ship9.ply	handgun7.ply

Figura 4.36: Matriz de Imagens padrão que possuem distância da soma menor do que 0,1.

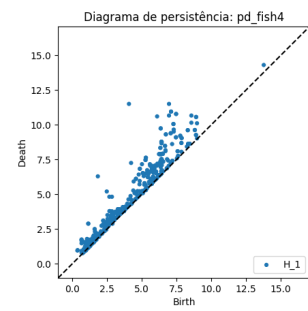
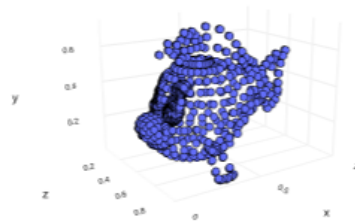
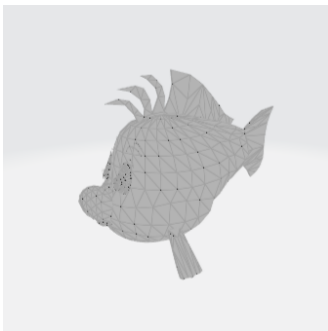
	A	B	C	D	E
1		1	2	3	4
2	desk_chair1.ply	flying_bird4.ply	sword0.ply	sword6.ply	sword8.ply
3	fish8.ply	guitar8.ply	ship2.ply		
4	flying_bird4.ply	desk_chair1.ply			
5	guitar8.ply	fish8.ply	ship2.ply		
6	ship2.ply	fish8.ply	guitar8.ply		
7	ship5.ply	helicopter0.ply			
8	sword0.ply	desk_chair1.ply	sword5.ply	sword6.ply	sword8.ply
9	sword5.ply	sword0.ply	sword6.ply	sword8.ply	
10	sword6.ply	desk_chair1.ply	sword0.ply	sword5.ply	sword8.ply
11	sword8.ply	desk_chair1.ply	sword0.ply	sword5.ply	sword6.ply
12	flying_bird7.ply	flying_bird6.ply			
13	helicopter0.ply	ship5.ply			
14	flying_bird6.ply	flying_bird7.ply			

Figura 4.37: Matriz de Imagens padrão que possuem distância do máximo menor do que 0,001.

Dentre as comparações realizadas, as imagens próximas à **Fish 2** da linha 3 da Figura 4.36 e à **Flying Bird 4** da linha 4 da Figura 4.37 foram geradas para observação.

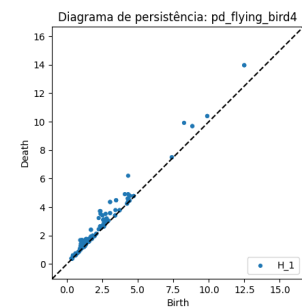
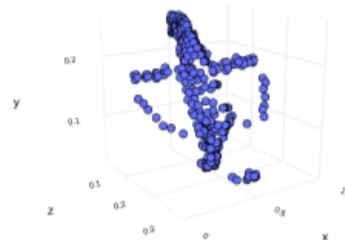


(a) Fish 2

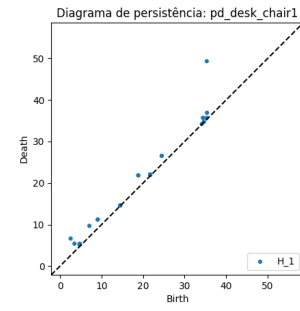
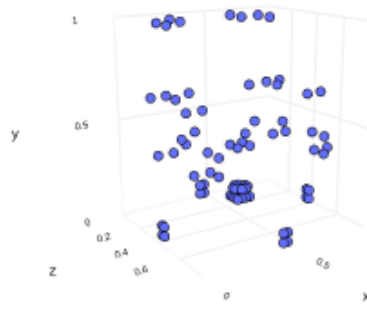


(b) Fish 4

Figura 4.39: Imagem próxima à Fish 2 - linha 3 Figura 4.36.



(a) Flying Bird 4



(b) Deskchair 1

Figura 4.41: Imagem próxima à Flying Bird 4 - linha 4 da Figura 4.37.

5 Resultados

No capítulo anterior, os dados foram preparados: gerados, organizados e filtrados e estão prontos para serem analisados.

Serão feitas três análises: o Escalonamento Multidimensional, a construção de Dendrogramas e a utilização do algoritmo *K*-means.

5.1 MDS - *Multidimensional Scaling*

A técnica do Escalonamento Multidimensional é usada para representar espacialmente, em 2D ou 3D, uma matriz de proximidades (semelhança ou dissimilhança) entre uma série de objetos de modo que possam ser mais facilmente visualizados.¹

Existem dois tipos principais de MDS: métrico (clássico) e não métrico. [10]

- **MDS métrico:** conhecido como Principal Component Analysis (PCA), tenta modelar a semelhança/diferença de dados calculando distâncias entre cada par de pontos usando suas coordenadas geométricas, medindo as distâncias em uma escala linear;
- **MDS não métrico:** foi projetado para lidar com dados ordinais, como classificação de preferência feita por clientes, por exemplo, onde a ordem importa e não seu valor absoluto.

Visto isto, a nossa análise será feita pelo método métrico, por meio das bibliotecas Python `sklearn.datasets - load_digits`, `sklearn - manifold` e `sklearn.manifold - MDS`.

O MDS métrico calcula distâncias entre cada par de pontos no espaço original de alta dimensão e transforma para um espaço de menor dimensão, preservando a distância entre os pontos da melhor forma possível. Este resultado é obtido pelo processo:

1. o algoritmo calcula distâncias entre cada par de pontos, d_{ij} ;
2. tenta otimizar a função custo (5.1), encontrando um conjunto de coordenadas em um espaço de menor dimensão que minimiza o valor do *Stress*.

$$\text{Stress}_D(x_1, \dots, x_n) = \sqrt{\sum_{i \neq j=1, \dots, n} (d_{ij} - \|x_i - x_j\|)^2}, \quad (5.1)$$

¹Disponível em: <https://igce.rc.unesp.br/Home/Departamentos47/geologiaaplicada/>. Acesso em: julho de 2022.

onde $x_1, \dots, x_n$ são os pontos com suas coordenadas de menor dimensão, d_{ij} é a distância original entre os pontos, e $\|x_i - x_j\|$ é a distância entre dois pontos correspondentes no espaço de dimensão menor. Quanto menor a diferença entre as distâncias, menor será o *Stress*, isto é, melhor será a otimização.

O algoritmo Python possibilita a inclusão de uma matriz distância, realizando apenas o item 2, anterior. Esta alternativa foi a escolhida para o trabalho, devido o banco de dados gerado nas subseções 4.4.1 e 4.5.

Como o banco de dados possui muitas imagens, de diferentes tipos, elas foram separadas por categorias e destacadas no gráfico segundo este agrupamento. Para isto, além da função que gera o escalonamento, no Programa 5.3, foram criadas outras duas para que esta classificação fosse possível, 5.1 e 5.2.

```

1  def dic_categoria(caminho_arquivo_nome):
2
3      arquivo = pd.read_csv(caminho_arquivo_nome, header=None)
4
5      n = len(arquivo.iloc[:, 0])
6
7      dicionario = {}
8
9      for i in range(0, n):
10         nome = arquivo.iloc[i, 0]
11         categoria = nome.split('.')[0][:-1]
12         if nome.startswith(categoria): # arq.começa_com('nome do arquivo
            ↳ ou letra')
13             if categoria in dicionario.keys():
14                 dicionario[categoria].append(nome)
15             else:
16                 dicionario[categoria] = [nome]
17
18     return dicionario

```

Programa 5.1: Função para separar os arquivos em categorias.

```

1  def coluna(caminho_categorias, caminho_arquivo, salvar=False,
2             caminho_salvar=''):
3
4      if salvar and caminho_salvar == '':
5          print('Para salvar é necessário escolher um caminho')
6          return None
7
8      arquivo = pd.read_csv(caminho_arquivo)
9      categorias = dic_categoria(caminho_categorias).keys()
10
11     lista_categorias = []
12
13     for i in arquivo.iloc[:, 0]:
14         nome = i.split('.')[0][:-1]
15         if nome in categorias:

```

```

16         lista_categorias.append(nome)
17
18     arquivo['Categorias'] = lista_categorias
19
20     return arquivo

```

Programa 5.2: Função para criar coluna de categorias nos csv's.

```

1  def MDS(caminho_arquivo, título, categorias=False,
2         caminho_nomes_arquivos='', salvar=False, caminho_salvar='',
3         tipo='png'):
4
5     csv = np.genfromtxt(caminho_arquivo, delimiter=',')[1:, 1:]
6     csv = csv*100
7
8     mds = manifold.MDS(n_components=2, metric=True, n_init=4,
9                       max_iter=300, verbose=0, n_jobs=None,
10                      random_state=None, dissimilarity='precomputed')
11
12     arquivo = mds.fit(csv).embedding_
13
14     if categorias:
15         categoria = coluna(caminho_nomes_arquivos, caminho_arquivo)
16         fig = px.scatter(categoria, arquivo[:, 0], arquivo[:, 1],
17                        color='Categorias', symbol='Categorias',
18                        template="simple_white", title=título)
19     else:
20         fig = px.scatter(csv, arquivo[:, 0], arquivo[:, 1],
21                        template="simple_white", title=título)
22     fig.update_xaxes(showgrid=False, showline=True, showdividers=False,
23                    showticklabels=False, title_font=dict(size=10))
24     fig.update_yaxes(showgrid=False, showline=True, showdividers=False,
25                    showticklabels=False, title_font=dict(size=10))
26     if salvar and tipo == 'png':
27         fig.write_image(caminho_salvar+'/MDS '+título+'.png')
28     if salvar and tipo == 'html':
29         fig.write_html(caminho_salvar+'/MDS '+título+'.html')
30     fig.show()

```

Programa 5.3: Função para gerar MDS de uma matriz de distâncias.

Obtendo as seguintes imagens para as matrizes de distância bottleneck, distância euclidiana, distância do máximo e distância da soma.

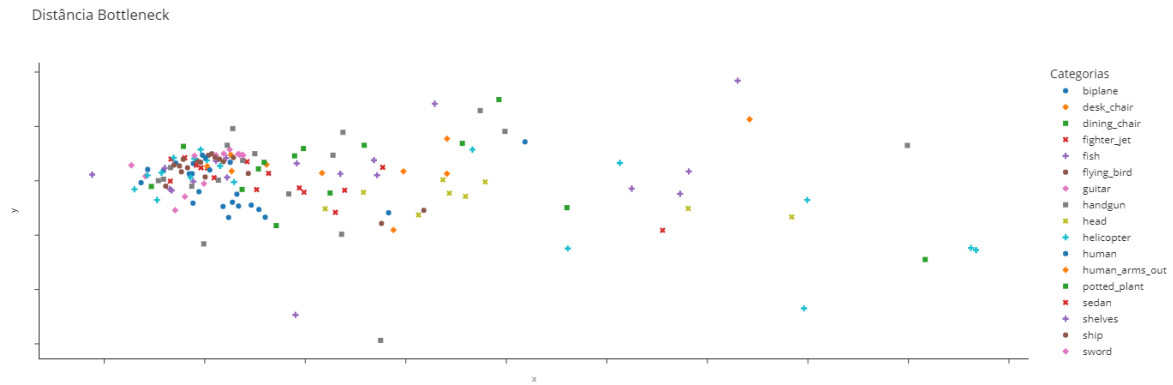


Figura 5.1: MDS distância bottleneck.

Como é possível perceber na imagem acima 5.1, devido a grande quantidade de imagens, a comparação pode ficar confusa. Para contornar este problema, a biblioteca `plotly.express` permite que seja feito uma espécie de filtro, onde os gráficos são readequados de acordo com a seleção das categorias presentes no lado direito do gráfico. Foram selecionadas as 5 primeiras categorias em todos os gráficos de distância. Os resultados, seguem 5.2, 5.3, 5.4 e 5.5.

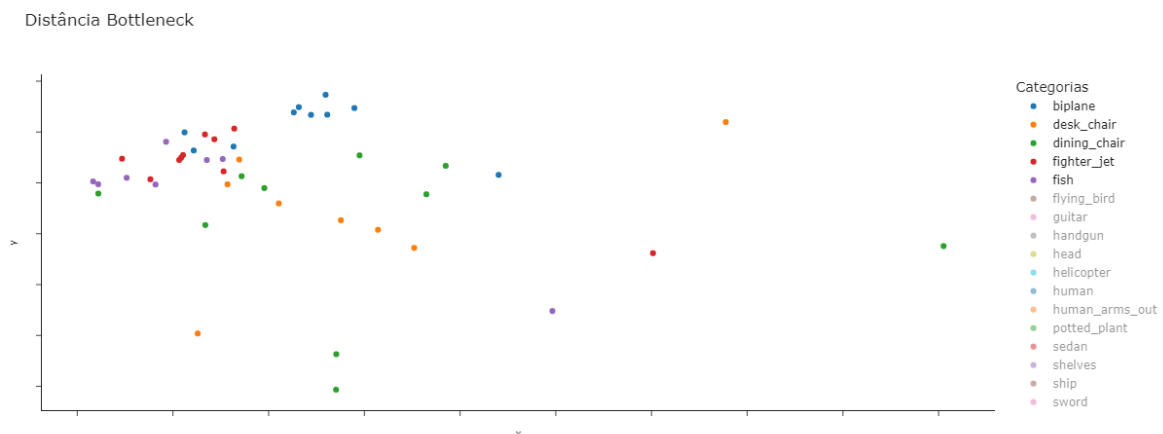


Figura 5.2: MDS distância bottleneck de 5 categorias.

Distância Euclidiana - peso

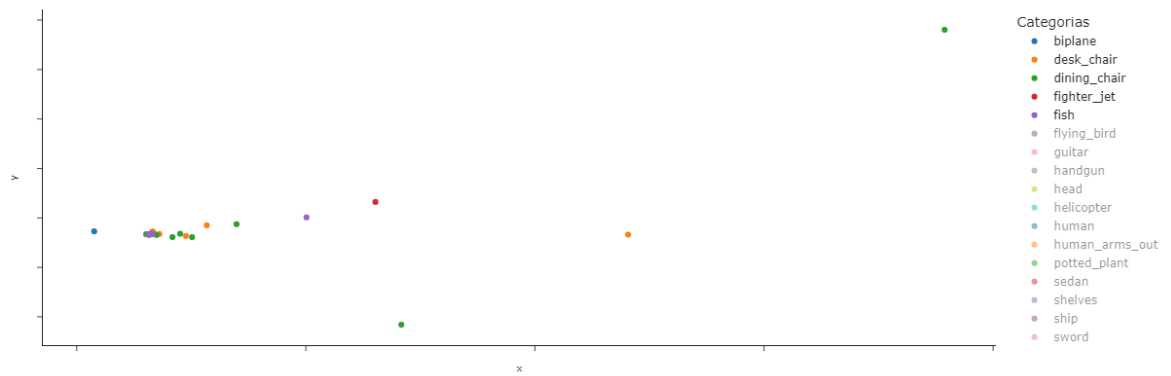


Figura 5.3: MDS distância euclidiana com peso de 5 categorias.

Distância do Máximo - peso

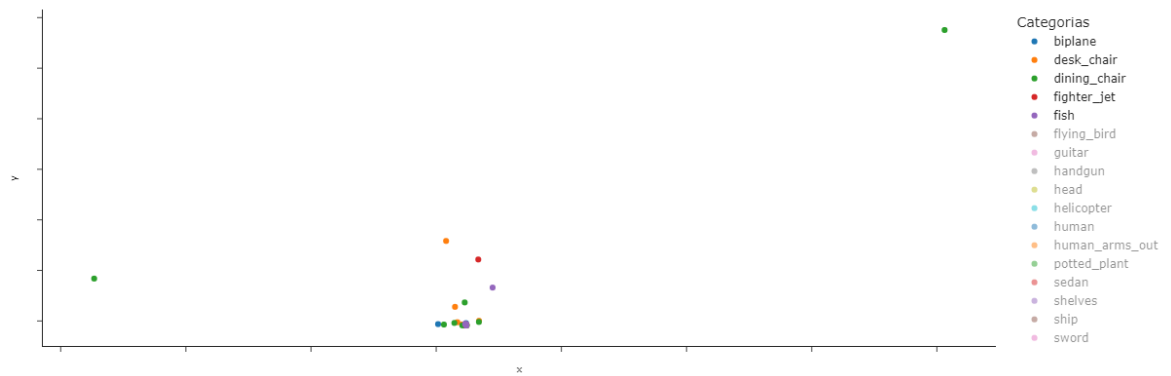


Figura 5.4: MDS distância do máximo com peso de 5 categorias.

Distância da Soma - peso

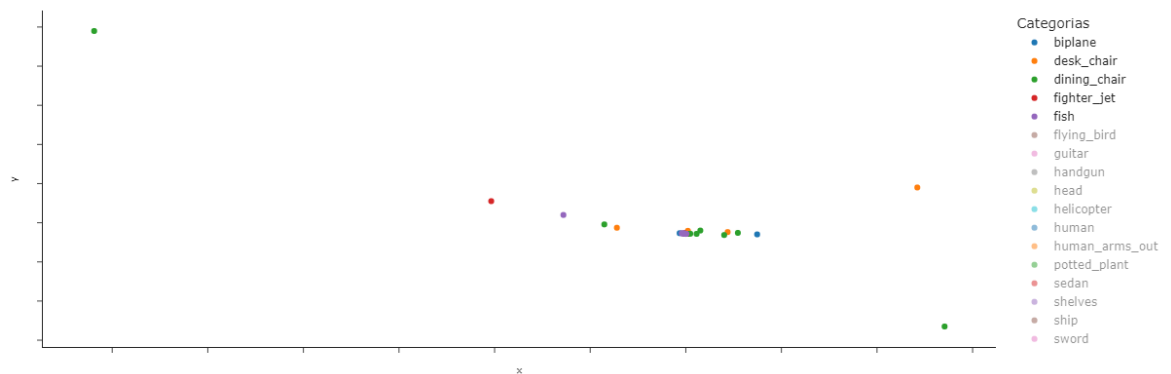


Figura 5.5: MDS distância da soma com peso de 5 categorias.

As métricas euclidiana, do máximo e da soma são equivalentes², assim, o comportamento do escalonamento entre as categorias é similar, como é possível observar nas imagens 5.3, 5.4 e 5.5. A igualdade não ocorre porque as medidas não são iguais e pela forma que a biblioteca `Python` gera o gráfico MDS - os pontos não são fixos nos eixos, o objetivo é mostrar a proximidade entre os objetos de estudo, segundo a distância informada. Devido a esta similaridade, basta que um único gráfico seja gerado. Dentre as três métricas, a da soma é a mais eficiente computacionalmente, como calculado durante a geração das imagens de persistência do capítulo anterior 4.5.3.

5.2 Dendrograma

Um dendrograma (dendro = árvore) é um tipo de diagrama que organiza determinados itens por meio de uma análise estatística de dados feita pela aplicação de um algoritmo de agrupamento, que resulta em coleções e sua ordenação hierárquica. Sua forma se assemelha a uma árvore, daí seu nome.³



Figura 5.6: Exemplo de dendrograma.

As linhas horizontais são chamadas **clades** ou **ramos**, e são organizados de acordo com a proximidade dos grupos, as linhas verticais são as **folhas**. Quanto maior a distância das linhas verticais no dendrograma, maior será a distância entre esses agrupamentos – esta é a base para a interpretação do gráfico. [21]

Os diagramas são criados seguindo a seguinte sequência:

1. Cada ponto de dados deve ser tratado como um grupo, no início;
2. Denote o número de grupos no início como K ;
3. Forme um grupo combinando os dois pontos de dados mais próximos, resultando em $K - 1$ grupos;
4. Forme mais grupos combinando os dois grupos mais próximos, resultando em $K - 2$ grupos;
5. Repita as etapas anteriores até que um único grande grupo seja criado.

Os grupos são construídos de duas maneiras:

- **Agrupamento divisivo:** é o método de agrupamento de cima para baixo, pois atribui todas as observações a um único agrupamento e, em seguida, particiona o agrupamento em dois agrupamentos menos semelhantes, de forma sucessiva.

²Demonstração disponível na página 24 da referência [2].

³Disponível em: <https://pt.wikipedia.org/wiki/Dendrograma>. Acesso em: julho de 2022.

- **Agrupamento aglomerativo:** é o método de agrupamento aglomerativo ou de baixo para cima, onde cada observação é atribuída ao seu próprio agrupamento.

Os algoritmos de agrupamento mais conhecidos são:

- **Ligação única:** geralmente produz grupos nos quais os indivíduos são adicionados sequencialmente a um único grupo. A distância entre os dois grupos é definida como a distância entre seus dois pontos de dados mais próximos;
- **Ligação completa:** geralmente produz grupos que são bem segregados e compactos. A distância entre os dois grupos é definida como a maior distância entre dois pontos de dados em cada grupo;
- **Média simples:** define a distância entre os grupos como a distância média entre cada um dos membros, ponderada de forma que os dois grupos tenham uma influência igual na saída final.

Para este trabalho, o agrupamento divisivo foi escolhido, utilizando um algoritmo descrito no Programa 5.4⁴ de média simples ward, podendo ser alterado para o algoritmo de ligação única single.

```

1 def dendrograma(caminho_arquivo, titulo, quantidade_niveis, truncar=False,
2                 salvar=False, caminho_salvar='', algoritmo='ward'):
3
4     if salvar and caminho_salvar == '':
5         print('Para salvar é necessário escolher um caminho')
6         return None
7
8     arquivo = np.genfromtxt(caminho_arquivo, delimiter=',')[1:, 1:]
9
10    if truncar:
11        plt.figure(figsize=(30, 10)) # figsize=(largura, altura)
12    else:
13        plt.figure(figsize=(15, 50)) # figsize=(largura, altura)
14    plt.title(titulo)
15    if algoritmo == 'single':
16        temp = hierarchy.linkage(arquivo, 'single') # linkage: faz os
17        ↪ cálculos do dendograma, single: algoritmo padrão - considera
18        ↪ as distâncias mais próximas
19    else:
20        temp = hierarchy.linkage(arquivo, 'ward') # ward: algoritmo que
21        ↪ considera a variância das distâncias do conjunto
22    if truncar:
23        dn = hierarchy.dendrogram(temp, truncate_mode='level',
24                                  p=quantidade_niveis) # truncate_mode:
25        ↪ limita o dendograma segundo a
26        ↪ quantidade de níveis escolhida
27    else:
28        dn = hierarchy.dendrogram(temp, orientation='right')

```

⁴A repetição da condicional if foi necessária devido a ordem de construção do gráfico, onde o tamanho deve ser o primeiro parâmetro, porém o dn depende do temp e não pôde ser incluído antes.

```

24     if salvar:
25         plt.savefig(caminho_salvar+'/Dendrograma '+titulo+'.png', dpi=300)
           ↪ # dpi: resolução
26     plt.show(dn)

```

Programa 5.4: Função para gerar dendrogramas de uma matriz de distâncias.

Os resultados para os dados deste trabalho foram gerados de duas maneiras: de forma truncada, onde são exibidos uma certa quantidade de níveis, e, exibindo todas as 158 imagens. Foram escolhidos 5 níveis para a forma truncada 5.7, 5.8, 5.9 e a forma horizontal para o dendrograma completo 5.10 e 5.11. Como apontado em 5.1, sobre métricas equivalentes, utilizamos apenas a matriz de distâncias da soma. Nos diagramas com alteração de peso, a coluna e linha referente à distância da figura `untitled3.ply` foi excluída, devido a grande discrepância que ela apresenta das demais.

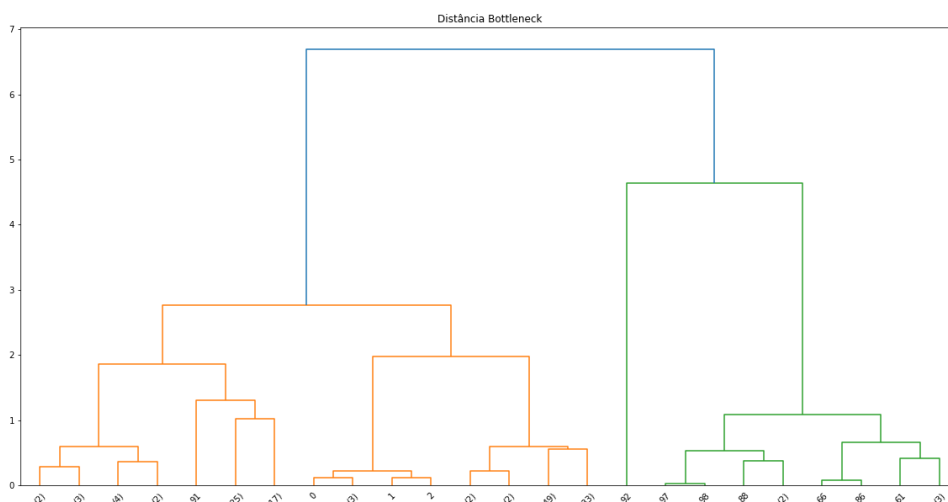
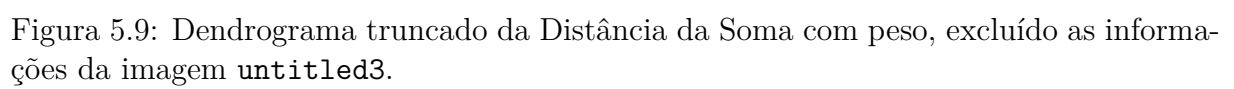
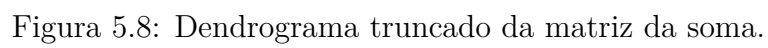


Figura 5.7: Dendrograma truncado da matriz de distância bottleneck.



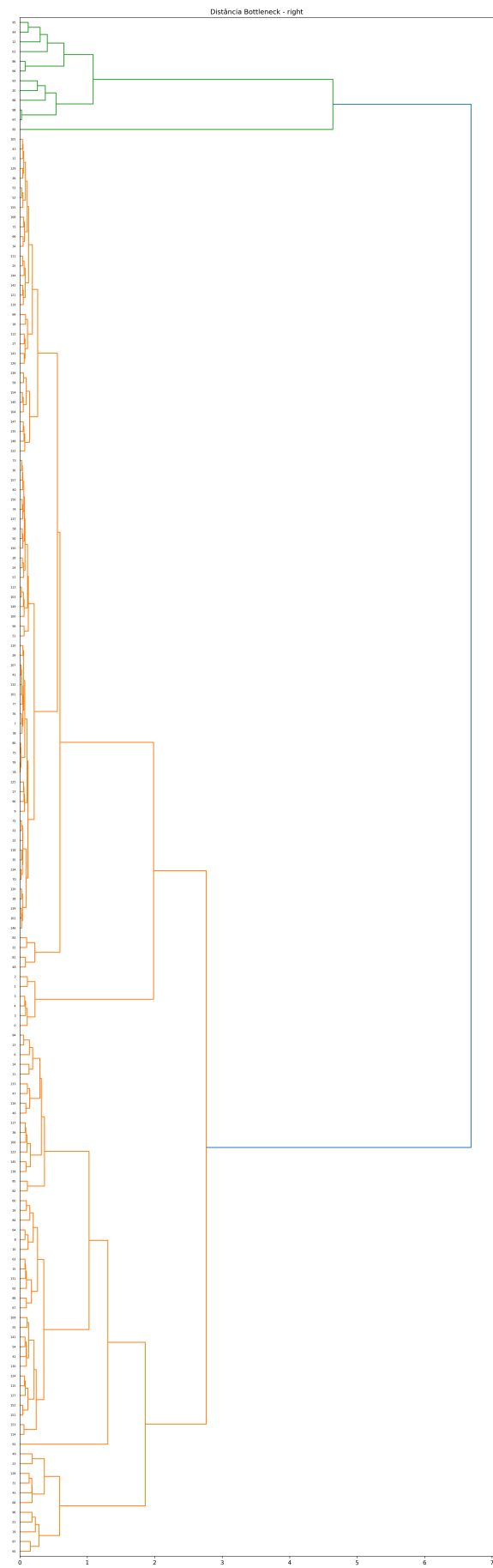


Figura 5.10: Dendrograma completo da Distância Bottleneck.

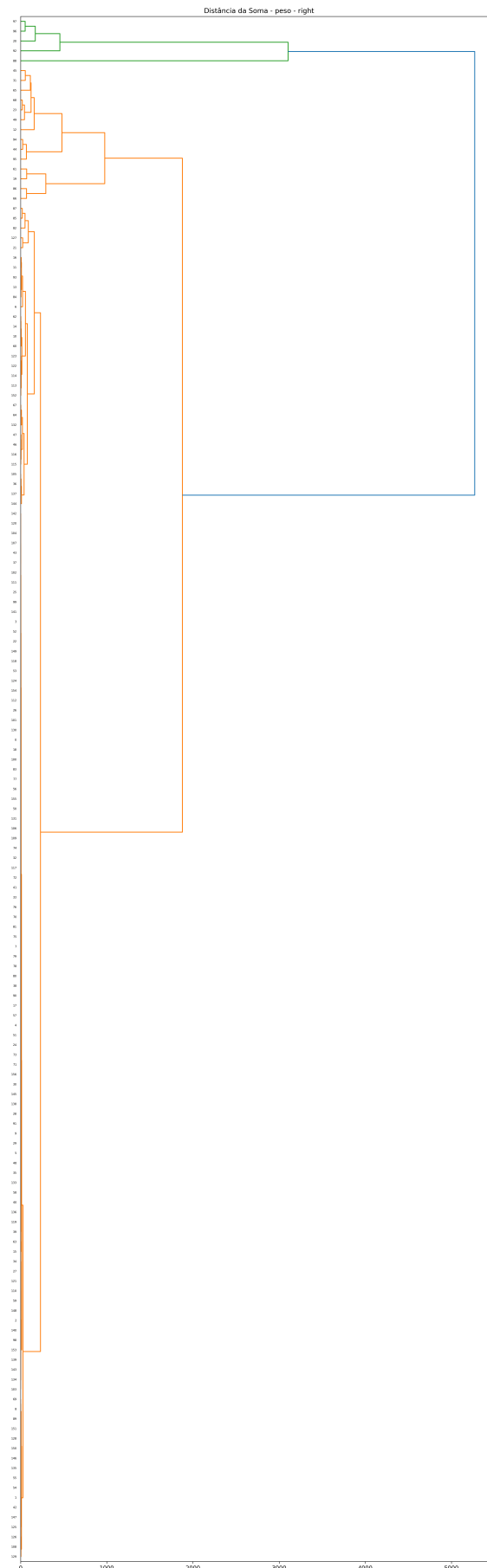


Figura 5.11: Dendrograma completo da Distância da Soma com peso, excluindo as informações da imagem `untitled3`.

Com o objetivo de melhorar a compreensão, já que o tamanho ideal para esta imagem seria maior do que esta página, na tabela abaixo 5.1 estão elencadas as imagens e seus respectivos números.

	Nome Arquivo	Categoria	Nome Arquivo (sem cubo)	Categoria (sem cubo)
1	biplane0.ply	biplane	biplane0.ply	biplane
2	biplane3.ply	biplane	biplane3.ply	biplane
3	biplane4.ply	biplane	biplane4.ply	biplane
4	biplane6.ply	biplane	biplane6.ply	biplane
5	biplane7.ply	biplane	biplane7.ply	biplane
6	biplane8.ply	biplane	biplane8.ply	biplane
7	biplane9.ply	biplane	biplane9.ply	biplane
8	desk_chair1.ply	desk_chair	desk_chair1.ply	desk_chair
9	desk_chair2.ply	desk_chair	desk_chair2.ply	desk_chair
10	desk_chair5.ply	desk_chair	desk_chair5.ply	desk_chair
11	desk_chair6.ply	desk_chair	desk_chair6.ply	desk_chair
12	desk_chair7.ply	desk_chair	desk_chair7.ply	desk_chair
13	desk_chair8.ply	desk_chair	desk_chair8.ply	desk_chair
14	dining_chair0.ply	dining_chair	dining_chair0.ply	dining_chair
15	dining_chair1.ply	dining_chair	dining_chair1.ply	dining_chair
16	dining_chair2.ply	dining_chair	dining_chair2.ply	dining_chair
17	dining_chair3.ply	dining_chair	dining_chair3.ply	dining_chair
18	dining_chair4.ply	dining_chair	dining_chair4.ply	dining_chair
19	dining_chair5.ply	dining_chair	dining_chair5.ply	dining_chair
20	dining_chair6.ply	dining_chair	dining_chair6.ply	dining_chair
21	dining_chair7.ply	dining_chair	dining_chair7.ply	dining_chair
22	dining_chair8.ply	dining_chair	dining_chair8.ply	dining_chair
23	dining_chair9.ply	dining_chair	dining_chair9.ply	dining_chair
24	fighter_jet0.ply	fighter_jet	fighter_jet0.ply	fighter_jet
25	fighter_jet3.ply	fighter_jet	fighter_jet3.ply	fighter_jet
26	fighter_jet4.ply	fighter_jet	fighter_jet4.ply	fighter_jet
27	fighter_jet5.ply	fighter_jet	fighter_jet5.ply	fighter_jet
28	fighter_jet6.ply	fighter_jet	fighter_jet6.ply	fighter_jet
29	fighter_jet7.ply	fighter_jet	fighter_jet7.ply	fighter_jet
30	fighter_jet8.ply	fighter_jet	fighter_jet8.ply	fighter_jet
31	fighter_jet9.ply	fighter_jet	fighter_jet9.ply	fighter_jet
32	fish3.ply	fish	fish3.ply	fish
33	fish5.ply	fish	fish5.ply	fish
34	fish8.ply	fish	fish8.ply	fish
35	fish9.ply	fish	fish9.ply	fish
36	flying_bird0.ply	flying_bird	flying_bird0.ply	flying_bird
37	flying_bird1.ply	flying_bird	flying_bird1.ply	flying_bird
38	flying_bird3.ply	flying_bird	flying_bird3.ply	flying_bird
39	flying_bird4.ply	flying_bird	flying_bird4.ply	flying_bird
40	flying_bird5.ply	flying_bird	flying_bird5.ply	flying_bird
41	flying_bird9.ply	flying_bird	flying_bird9.ply	flying_bird
42	guitar8.ply	guitar	guitar8.ply	guitar

	Nome Arquivo	Categoria	Nome Arquivo (sem cubo)	Categoria (sem cubo)
43	handgun5.ply	handgun	handgun5.ply	handgun
44	handgun7.ply	handgun	handgun7.ply	handgun
45	head3.ply	head	head3.ply	head
46	head6.ply	head	head6.ply	head
47	head7.ply	head	head7.ply	head
48	head8.ply	head	head8.ply	head
49	helicopter4.ply	helicopter	helicopter4.ply	helicopter
50	helicopter7.ply	helicopter	helicopter7.ply	helicopter
51	helicopter8.ply	helicopter	helicopter8.ply	helicopter
52	human0.ply	human	human0.ply	human
53	human1.ply	human	human1.ply	human
54	human2.ply	human	human2.ply	human
55	human5.ply	human	human5.ply	human
56	human7.ply	human	human7.ply	human
57	human8.ply	human	human8.ply	human
58	human_arms_ out0.ply	human_arms_ out	human_arms_ out0.ply	human_arms_ out
59	human_arms_ out7.ply	human_arms_ out	human_arms_ out7.ply	human_arms_ out
60	potted_plant9.ply	potted_plant	potted_plant9.ply	potted_plant
61	sedan2.ply	sedan	sedan2.ply	sedan
62	shelves0.ply	shelves	shelves0.ply	shelves
63	shelves2.ply	shelves	shelves2.ply	shelves
64	shelves3.ply	shelves	shelves3.ply	shelves
65	shelves4.ply	shelves	shelves4.ply	shelves
66	shelves5.ply	shelves	shelves5.ply	shelves
67	shelves6.ply	shelves	shelves6.ply	shelves
68	shelves7.ply	shelves	shelves7.ply	shelves
69	shelves8.ply	shelves	shelves8.ply	shelves
70	shelves9.ply	shelves	shelves9.ply	shelves
71	ship0.ply	ship	ship0.ply	ship
72	ship1.ply	ship	ship1.ply	ship
73	ship2.ply	ship	ship2.ply	ship
74	ship4.ply	ship	ship4.ply	ship
75	ship5.ply	ship	ship5.ply	ship
76	sword0.ply	sword	sword0.ply	sword
77	sword1.ply	sword	sword1.ply	sword
78	sword3.ply	sword	sword3.ply	sword
79	sword5.ply	sword	sword5.ply	sword
80	sword6.ply	sword	sword6.ply	sword
81	sword7.ply	sword	sword7.ply	sword
82	sword8.ply	sword	sword8.ply	sword
83	table0.ply	table	table0.ply	table
84	table1.ply	table	table1.ply	table
85	table2.ply	table	table2.ply	table
86	table3.ply	table	table3.ply	table

	Nome Arquivo	Categoria	Nome Arquivo (sem cubo)	Categoria (sem cubo)
87	table4.ply	table	table4.ply	table
88	table5.ply	table	table5.ply	table
89	table6.ply	table	table6.ply	table
90	table7.ply	table	table7.ply	table
91	table8.ply	table	table8.ply	table
92	table9.ply	table	table9.ply	table
93	untitled3.ply	untitled	vase0.ply	vase
94	vase0.ply	vase	vase3.ply	vase
95	vase3.ply	vase	vase4.ply	vase
96	vase4.ply	vase	vase5.ply	vase
97	vase5.ply	vase	vase7.ply	vase
98	vase7.ply	vase	vase8.ply	vase
99	vase8.ply	vase	vase9.ply	vase
100	vase9.ply	vase	biplane1.ply	biplane
101	biplane1.ply	biplane	biplane2.ply	biplane
102	biplane2.ply	biplane	fighter_jet2.ply	fighter_jet
103	fighter_jet2.ply	fighter_jet	fish2.ply	fish
104	fish2.ply	fish	fish4.ply	fish
105	fish4.ply	fish	fish6.ply	fish
106	fish6.ply	fish	flying_bird2.ply	flying_bird
107	flying_bird2.ply	flying_bird	flying_bird7.ply	flying_bird
108	flying_bird7.ply	flying_bird	guitar7.ply	guitar
109	guitar7.ply	guitar	guitar9.ply	guitar
110	guitar9.ply	guitar	handgun0.ply	handgun
111	handgun0.ply	handgun	handgun1.ply	handgun
112	handgun1.ply	handgun	handgun3.ply	handgun
113	handgun3.ply	handgun	handgun6.ply	handgun
114	handgun6.ply	handgun	head0.ply	head
115	head0.ply	head	head1.ply	head
116	head1.ply	head	head2.ply	head
117	head2.ply	head	head9.ply	head
118	head9.ply	head	helicopter0.ply	helicopter
119	helicopter0.ply	helicopter	helicopter2.ply	helicopter
120	helicopter2.ply	helicopter	helicopter3.ply	helicopter
121	helicopter3.ply	helicopter	helicopter9.ply	helicopter
122	helicopter9.ply	helicopter	human4.ply	human
123	human4.ply	human	human9.ply	human
124	human9.ply	human	potted_plant0.ply	potted_plant
125	potted_plant0.ply	potted_plant	potted_plant8.ply	potted_plant
126	potted_plant8.ply	potted_plant	race_car2.ply	race_car
127	race_car2.ply	race_car	sedan3.ply	sedan
128	sedan3.ply	sedan	shelves1.ply	shelves
129	shelves1.ply	shelves	vase1.ply	vase
130	vase1.ply	vase	vase2.ply	vase
131	vase2.ply	vase	desk_chair0.ply	desk_chair
132	desk_chair0.ply	desk_chair	flying_bird6.ply	flying_bird

	Nome Arquivo	Categoria	Nome Arquivo (sem cubo)	Categoria (sem cubo)
133	flying_bird6.ply	flying_bird	head5.ply	head
134	head5.ply	head	human3.ply	human
135	human3.ply	human	sedan7.ply	sedan
136	sedan7.ply	sedan	vase6.ply	vase
137	vase6.ply	vase	biplane5.ply	biplane
138	biplane5.ply	biplane	desk_chair3.ply	desk_chair
139	desk_chair3.ply	desk_chair	fighter_jet1.ply	fighter_jet
140	fighter_jet1.ply	fighter_jet	fish7.ply	fish
141	fish7.ply	fish	guitar0.ply	guitar
142	guitar0.ply	guitar	guitar1.ply	guitar
143	guitar1.ply	guitar	guitar5.ply	guitar
144	guitar5.ply	guitar	handgun2.ply	handgun
145	handgun2.ply	handgun	handgun8.ply	handgun
146	handgun8.ply	handgun	helicopter6.ply	helicopter
147	helicopter6.ply	helicopter	potted_plant1.ply	potted_plant
148	potted_plant1.ply	potted_plant	race_car0.ply	race_car
149	race_car0.ply	race_car	race_car4.ply	race_car
150	race_car4.ply	race_car	race_car6.ply	race_car
151	race_car6.ply	race_car	sedan0.ply	sedan
152	sedan0.ply	sedan	sedan1.ply	sedan
153	sedan1.ply	sedan	sedan6.ply	sedan
154	sedan6.ply	sedan	ship3.ply	ship
155	ship3.ply	ship	ship6.ply	ship
156	ship6.ply	ship	ship8.ply	ship
157	ship8.ply	ship	ship9.ply	ship
158	ship9.ply	ship		

Tabela 5.1: Legenda para a numeração dos dendrogramas.

5.2.1 K -means

K -Means é um algoritmo de agrupamento, que avalia e agrupa os dados de acordo com suas características. [1]

A ideia lógica deste agrupamento é a seguinte:

1. Define o número de K agrupamentos;
2. Define uma amostra para cada grupo, chamado **centróide**, feito de forma aleatória;
3. Calcula, para cada ponto, o centróide mais próximo e agrupa;
4. Escolhe um novo centro para este novo grupo. A escolha é feita pela média da posição dos itens pertencentes a este grupo.

Os itens 3 e 4 são feitos de forma iterativa até que todas as amostras estejam agrupadas na quantidade de grupos escolhida.

O Programa 5.5 é simples e retorna um vetor, onde cada posição deste vetor indica a que grupo o item pertence. A quantidade de 4 grupos foi escolhida pela observação do número de folhas grandes dos Dendrogramas.

```

1 def kmeans(caminho_arquivo, numero_cluster):
2
3     arquivo = np.genfromtxt(caminho_arquivo, delimiter=',')[1:, 1:]
4
5     kmeans = KMeans(n_clusters=numero_cluster, init='k-means++',
6                     max_iter=400, n_init=20)
7     grupos = kmeans.fit_predict(arquivo) # computa o KMeans e retorna os
      ↪ conjuntos que cada amostra pertence
8
9     return grupos

```

Programa 5.5: Função para calcular o agrupamento K -means.

A tabela 5.2 compara as divisões feitas pelo algoritmo para a Distância Bottleneck e para Distância da Soma com peso.

Nome Arquivo	Distância Bottleneck	Distância Soma (peso)	Categoria
biplane0.ply	2	2	biplane
biplane3.ply	2	2	biplane
biplane4.ply	2	2	biplane
biplane6.ply	2	2	biplane
biplane7.ply	2	2	biplane
biplane8.ply	2	2	biplane
biplane9.ply	0	2	biplane
desk_chair1.ply	2	2	desk_chair
desk_chair2.ply	2	2	desk_chair
desk_chair5.ply	2	2	desk_chair
desk_chair6.ply	0	2	desk_chair
desk_chair7.ply	0	2	desk_chair
desk_chair8.ply	1	2	desk_chair
dining_chair0.ply	0	2	dining_chair
dining_chair1.ply	0	2	dining_chair
dining_chair2.ply	2	2	dining_chair
dining_chair3.ply	2	2	dining_chair
dining_chair4.ply	2	2	dining_chair
dining_chair5.ply	2	2	dining_chair
dining_chair6.ply	0	0	dining_chair
dining_chair7.ply	1	0	dining_chair
dining_chair8.ply	0	2	dining_chair
dining_chair9.ply	2	2	dining_chair
fighter_jet0.ply	0	2	fighter_jet
fighter_jet3.ply	2	2	fighter_jet
fighter_jet4.ply	2	2	fighter_jet
fighter_jet5.ply	2	2	fighter_jet
fighter_jet6.ply	2	2	fighter_jet
fighter_jet7.ply	2	2	fighter_jet
fighter_jet8.ply	2	2	fighter_jet
fighter_jet9.ply	2	2	fighter_jet

Nome Arquivo	Distância Bottleneck	Distância Soma (peso)	Categoria
fish3.ply	0	2	fish
fish5.ply	2	2	fish
fish8.ply	2	2	fish
fish9.ply	2	2	fish
flying_bird0.ply	2	2	flying_bird
flying_bird1.ply	0	2	flying_bird
flying_bird3.ply	2	2	flying_bird
flying_bird4.ply	2	2	flying_bird
flying_bird5.ply	2	2	flying_bird
flying_bird9.ply	2	2	flying_bird
guitar8.ply	2	2	guitar
handgun5.ply	2	2	handgun
handgun7.ply	2	2	handgun
head3.ply	1	2	head
head6.ply	0	2	head
head7.ply	0	2	head
head8.ply	0	2	head
helicopter4.ply	2	2	helicopter
helicopter7.ply	0	2	helicopter
helicopter8.ply	2	2	helicopter
human0.ply	2	2	human
human1.ply	2	2	human
human2.ply	2	2	human
human5.ply	2	2	human
human7.ply	2	2	human
human8.ply	2	2	human
human_arms_out0.ply	2	2	human_arms_out
human_arms_out7.ply	2	2	human_arms_out
potted_plant9.ply	2	2	potted_plant
sedan2.ply	2	2	sedan
shelves0.ply	1	0	shelves
shelves2.ply	2	2	shelves
shelves3.ply	2	2	shelves
shelves4.ply	2	2	shelves
shelves5.ply	0	2	shelves
shelves6.ply	1	0	shelves
shelves7.ply	2	2	shelves
shelves8.ply	0	2	shelves
shelves9.ply	2	2	shelves
ship0.ply	2	2	ship
ship1.ply	2	2	ship
ship2.ply	2	2	ship
ship4.ply	2	2	ship
ship5.ply	2	2	ship
sword0.ply	2	2	sword
sword1.ply	2	2	sword

Nome Arquivo	Distância Bottleneck	Distância Soma (peso)	Categoria
sword3.ply	2	2	sword
sword5.ply	2	2	sword
sword6.ply	2	2	sword
sword7.ply	2	2	sword
sword8.ply	2	2	sword
table0.ply	0	2	table
table1.ply	2	2	table
table2.ply	0	2	table
table3.ply	0	2	table
table4.ply	1	0	table
table5.ply	0	2	table
table6.ply	1	3	table
table7.ply	2	2	table
table8.ply	2	2	table
table9.ply	2	2	table
untitled3.ply	3	1	untitled
vase0.ply	1	0	vase
vase3.ply	0	2	vase
vase4.ply	1	2	vase
vase5.ply	0	2	vase
vase7.ply	1	0	vase
vase8.ply	1	0	vase
vase9.ply	2	2	vase
biplane1.ply	2	2	biplane
biplane2.ply	2	2	biplane
fighter_jet2.ply	2	2	fighter_jet
fish2.ply	2	2	fish
fish4.ply	2	2	fish
fish6.ply	2	2	fish
flying_bird2.ply	0	2	flying_bird
flying_bird7.ply	2	2	flying_bird
guitar7.ply	2	2	guitar
guitar9.ply	2	2	guitar
handgun0.ply	2	2	handgun
handgun1.ply	2	2	handgun
handgun3.ply	2	2	handgun
handgun6.ply	2	2	handgun
head0.ply	2	2	head
head1.ply	2	2	head
head2.ply	0	2	head
head9.ply	0	2	head
helicopter0.ply	2	2	helicopter
helicopter2.ply	2	2	helicopter
helicopter3.ply	2	2	helicopter
helicopter9.ply	2	2	helicopter
human4.ply	2	2	human

Nome Arquivo	Distância Bottleneck	Distância Soma (peso)	Categoria
human9.ply	0	2	human
potted_plant0.ply	2	2	potted_plant
potted_plant8.ply	2	2	potted_plant
race_car2.ply	2	2	race_car
sedan3.ply	2	2	sedan
shelves1.ply	0	2	shelves
vase1.ply	2	2	vase
vase2.ply	2	2	vase
desk_chair0.ply	2	2	desk_chair
flying_bird6.ply	2	2	flying_bird
head5.ply	0	2	head
human3.ply	2	2	human
sedan7.ply	2	2	sedan
vase6.ply	2	2	vase
biplane5.ply	2	2	biplane
desk_chair3.ply	0	2	desk_chair
fighter_jet1.ply	2	2	fighter_jet
fish7.ply	2	2	fish
guitar0.ply	2	2	guitar
guitar1.ply	2	2	guitar
guitar5.ply	2	2	guitar
handgun2.ply	2	2	handgun
handgun8.ply	0	2	handgun
helicopter6.ply	2	2	helicopter
potted_plant1.ply	2	2	potted_plant
race_car0.ply	2	2	race_car
race_car4.ply	2	2	race_car
race_car6.ply	2	2	race_car
sedan0.ply	2	2	sedan
sedan1.ply	2	2	sedan
sedan6.ply	0	2	sedan
ship3.ply	2	2	ship
ship6.ply	2	2	ship
ship8.ply	2	2	ship
ship9.ply	2	2	ship

Tabela 5.2: Tabela Comparativa de Resultado do agrupamento K -means.

6 Conclusão

Os resultados gerados no último capítulo, apresentaram algumas características interessantes do banco de imagens utilizado no trabalho.

A filtração do escalonamento possibilitou notar que, mesmo pertencendo à uma mesma categoria, algumas imagens apresentaram um distanciamento das demais. Tratando estes pontos como isolados, percebe-se uma aproximação das imagens de mesma categoria. Na Figura 5.2, fica clara a proximidade entre as bolinhas vermelhas, entre as azuis e certo distanciamento entre elas. Utilizando a métrica da soma 5.5, a proximidade entre os pontos é ainda maior.

O Dendrograma é uma ferramenta muito utilizada na clusterização, porém, devido sua extensão, neste caso, tornou a análise difícil. Porém, ao focar nas folhas verdes, constata-se resultados parecidos entre a Distância Bottleneck e as Imagens de Persistência, utilizando a distância da soma, com peso. Isso mostra que bons resultados podem ser obtidos ao se trabalhar com as imagens, que apresentaram uma eficiência computacional muito superior.

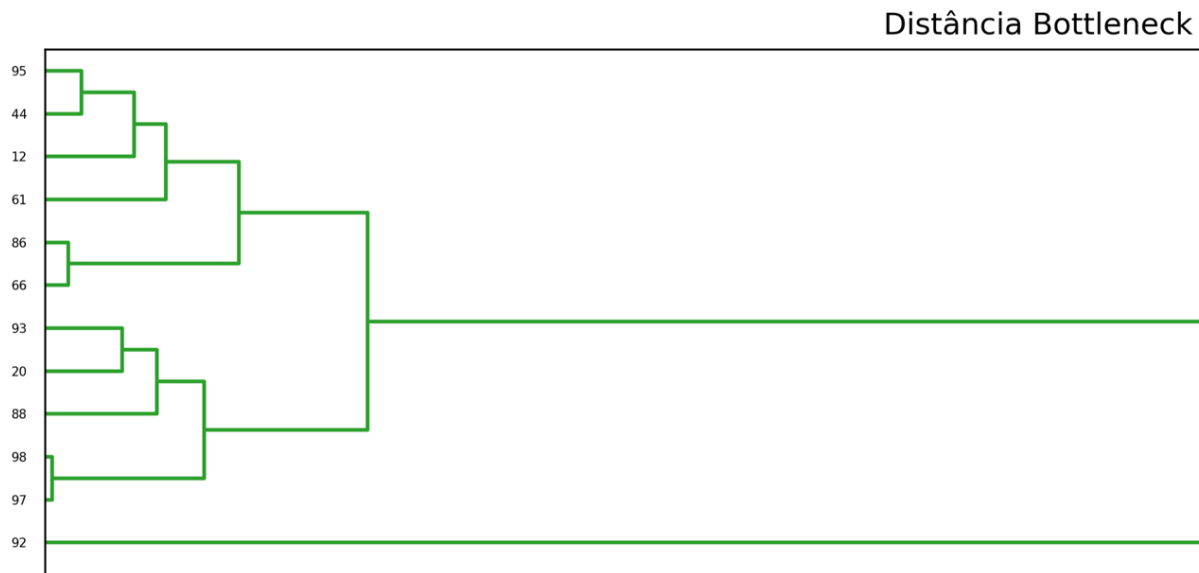


Figura 6.1: Folhas verdes do dendrograma 5.10.

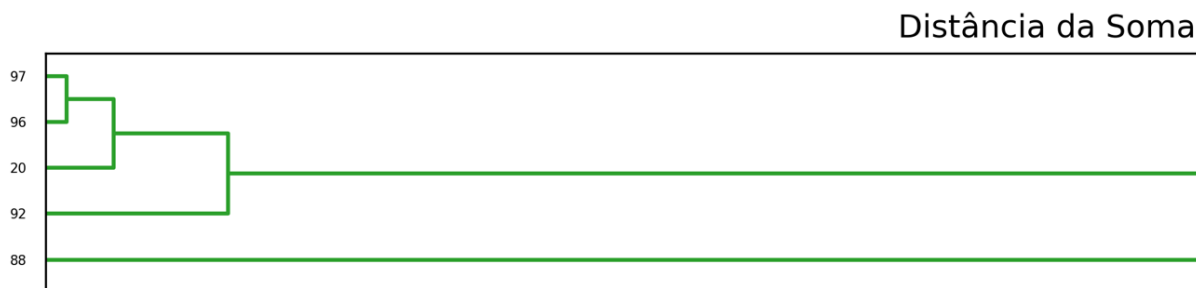


Figura 6.2: Folhas verdes do dendrograma 5.11.

As Figuras 6.1 e 6.2 mostram que 20 (Dining_chair), 88 (Table), 92 (Table) e 97 (Vase), pertencem a ramos próximos.

A ferramenta *K-means* agrupou várias categorias em um único conjunto, entretanto, imagens de uma mesma categoria, estão juntas, o que acorda com o esperado. Quando executado, este programa representa os grupos de forma aleatória, assim, é preciso uma observação mais cuidadosa para poder comparar as duas distâncias. Os grupos maiores foram representados igualmente por 2, o 0 da soma refere-se ao grupo 1 da bottleneck, o 1 da soma refere-se ao grupo 3 da bottleneck e por fim, o 3 da soma, refere-se ao grupo 0 da bottleneck. Um fato que ajudou a construir esta relação foram as quantidades por grupos, como mostra a tabela 6.1 e o fato do grupo 3 da bottleneck e 1 da soma conterem apenas a imagem `untitled3`.

Grupo	Bottleneck	Soma (peso)
0	31	8
1	11	1
2	115	148
3	1	1

Tabela 6.1: Quantidade de imagem em cada grupo

De forma geral, os resultados obtidos neste trabalho mostraram que para imagens muito heterogêneas, não há uma conclusão clara, apesar das possíveis interpretações.

O banco de dados utilizado para desenvolver a pesquisa foi de tamanho desmedido, com imagens de diversos tamanhos (como foi possível observar na escala das figuras utilizadas no decorrer das páginas), de categorias muito distintas e um intervalo de número de pontos muito abrangente. E sem haver um estudo anterior sobre estas imagens, não é possível constatar se os objetivos foram, de fato, alcançados. Porém, foram promissores.

A vantagem de utilizar a topologia como ferramenta para este tipo de análise, é que a orientação do objeto não importa. O tamanho influenciou no estudo das Imagens de Persistência, porém não foi tão importante para a distância Bottleneck. A distância entre os pontos intervém nos intervalos de persistência, logo, aparenta ser o principal motivo das discrepâncias no resultados.

Referências

- [1] ANASTACIO, Bruno. **K-means: o que é, como funciona, aplicações e exemplo em Python**. Maio, 2020. Disponível em: [https://medium.com/programadores\[...\]/](https://medium.com/programadores[...]/). Acesso em: julho de 2022.
- [2] ANDRADE, Maria Gorete Carreira. **Notas de aula - Topologia 1**. Março de 2021. Ibilce - Unesp.
- [3] CERDA, Maurício; FRÍAS, José; HITSCHFEDL, Nancy; KINDELAN, Rolando. Classification based on topological data analysis. **A Preprint**, 09 de Fevereiro de 2021.
- [4] CHEPUSHTANOVA, Sofya. et al. Persistence Images: An Alternative Persistent Homology Representation. 22 de Julho de 2015.
- [5] COLABORATORY. Perguntas Frequentes. Google, [s.d.]. Disponível em: <https://research.google.com/colaboratory/>. Acesso em: março de 2022.
- [6] CORTELLA, Mario Sergio. **Pensador**. [—]. Disponível em: <https://www.pensador.com/frase/MTQzMTA1Ng/>. Acesso em: junho de 2022.
- [7] **COS 429 - Computer Vision**. Princeton University. Princeton, Nova Jérsei, 2005. Disponível em: <https://www.cs.princeton.edu/courses/>. Acesso em: novembro de 2021.
- [8] CRIAR ou editar arquivos .csv para importação para o Outlook. **Microsoft**, [s.d.]. Disponível em: <https://support.microsoft.com/pt-br/office/>. Acesso em: março de 2022.
- [9] CROOM, Fred; **Basic Concepts of Algebraic Topology**. Springer-Verlag, New York Inc, 1978.
- [10] DOBILAS, Saul. MDS: Multidimensional Scaling — Smart Way to Reduce Dimensionality in Python. **Towards Data Science**, 11 de Setembro de 2021. Disponível em: <https://towardsdatascience.com/mds>. Acesso em: julho de 2022.
- [11] EDELSBRUNNER, Hebert; HARER, John L. **Computation Topology: An Introduction**. American Mathematical Society - AMS. Michigan, USA, 2010.
- [12] EXTENSÃO do arquivo PLY. **File.extension**, 14 de abril de 2020. Disponível em: <https://www.file-extension.info/pt/format/ply>. Acesso em: março de 2022.

-
- [13] FARIAS, Ana Maria Lima de. **Variáveis Aleatórias Bidimensionais**. Universidade Federal Fluminense. [—].
- [14] MELO, Thiago de; SANTOS, Richard G. dos. Homologia Persistente: um breve estudo da distância Bottleneck no agrupamento de imagens. **SBMAC**, 2021.
- [15] MENEGON, Aurélio. Notas de aula - Introdução à topologia algébrica. **Escola de Verão 2021**, Universidade Federal da Paraíba. Paraíba, 2021.
- [16] MUNKRES, James. **Elements of algebraic topology**. The Benjamin/Cummings Publishing Company, Inc. Califórnia, 1930.
- [17] O QUE É BLENDER? **Oficina da net**, 16 de março de 2010. Disponível em: https://www.oficinadanet.com.br/artigo/criacao_3d/blender_o_que_e. Acesso em: março de 2022.
- [18] PINTO, Guilherme Vituri Fernandes. **Topological Data Analysis: uma visão geral**. Unesp - IBILCE. [—].
- [19] REN, Tsang Ing. Notas de aula - Processamento de imagens coloridas. **Centro de Informática**. Universidade Federal de Pernambuco - UFPE. [—].
- [20] RONCHI, Carlos Henrique Venturi. **Geradores de homologia persistente e aplicações**. Dissertação (Mestrado) - Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2020.
- [21] SCIPY - DENDROGRAMA DE HIERARQUIA DE CLUSTER. **Acervo Lima**. Disponível em: [https://acervolima.com/scipy-dendrograma\[...\]/](https://acervolima.com/scipy-dendrograma[...]/). Acesso em: julho de 2022.
- [22] SILVA, Fernando Gasparotto. **Estrutura e estabilidade de módulos de persistência**. Dissertação (Mestrado) - Instituto de Biociências, Letras e Ciências Exatas, Universidade Estadual Paulista “Júlio de Mesquita Filho”, São José do Rio Preto, 2017.
- [23] SOUZA, Mariele Pedro. Homologia Simplicial. **Relatório de Iniciação Científica**. Unesp - IBILCE, 2019.

A Lista de bibliotecas utilizadas na pesquisa

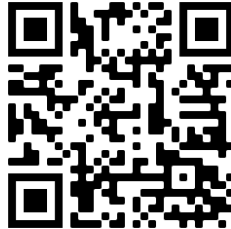
```
1 from itertools import product
2
3 import kaleido # salvar html
4
5 import numpy as np # vetores, matrizes e tabelas
6
7 import matplotlib.pyplot as plt # gráficos
8
9 import os # pastas
10
11 import open3d as o3d # arquivos tridimensionais
12
13 import pandas as pd # tabelas
14
15 import plotly.express as px # nuvem de pontos html
16 import plotly.graph_objects as go # imagens html
17
18 import persim # criar diagramas de persistência
19 from persim import PersistenceImager
20
21 from ripser import Rips # criar filtração
22
23 from scipy.cluster import hierarchy
24 import scipy.spatial.distance as ssd
25
26 from sklearn import manifold
27 from sklearn.cluster import AgglomerativeClustering # dendrograma
28 from sklearn.cluster import KMeans # dendrograma
29 from sklearn.manifold import MDS # MDS
30
31 import tadatasets
32
33 import time # tempo
34
35 import yaml # gerar gráfico 3d editável
```

A.1 Documentação das Bibliotecas

Abaixo, é possível acessar informações sobre cada uma das bibliotecas Python listadas. O acesso pode ser feito pelo link ou por meio do QR-Code.



(a) `itertools`



(b) `kaleido`



(c) `numpy`



(d) `matplotlib`



(e) `os`



(f) `open3d`



(g) `pandas`



(h) `plotly`



(i) `persim`



(j) `riper`



(k) `scipy`



(l) `sklearn`



(m) `tadatasets`



(n) `time`



(o) `yaml`

B QR-Codes

B.1 Notas de Rodapé

Esta seção traz, em forma de QR-Codes, os links das notas de rodapé presentes no texto, separados por capítulos.

B.1.1 Agradecimentos



(a) Cora Coralina, pg. 7

B.1.2 Capítulo 1



(a) Lisa Piccorillo, pg. 21 (b) Hiperplano, pg. 21 (c) Coordenadas bari-cêntricas, pg. 22 (d) Triângulo de Pascal, pg. 24

B.1.3 Capítulo 2



(a) Espaço Vetoriais,
pg. 36

B.1.4 Capítulo 3



(a) Métrica, pg. 64

B.1.5 Capítulo 4



(a) MDS, pg. 83



(b) Dendrograma, pg. 88

B.2 Referências Bibliográficas



(a) [1]



(b) [5]



(c) [8]



(d) [7]



(e) [10]



(f) [12]



(g) [17]



(h) [21]

B.3 Site

As imagens utilizadas na pesquisa estão disponíveis para consulta em um site, criado para tal. O acesso pode ser feito pelo link da descrição da imagem abaixo ou pelo código QR.

(a) [Site Dissertação](#)